RWTH Aachen

Department of Computer Science
Technical Report

Reduction Techniques for Nondeterministic and Probabilistic Systems

Arpit Sharma

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2015-03

RWTH Aachen · Department of Computer Science · January 2015

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Reduction Techniques for Nondeterministic and Probabilistic Systems

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Arpit Sharma, MSc

aus

Pilani, India

Berichter: Prof. Dr. Ir. Joost-Pieter Katoen

Prof. Dr. Holger Hermanns

Tag der mündlichen Prüfung: 15. January 2015

Arpit Sharma
Lehrstuhl für Informatik 2
arpit.sharma@cs.rwth-aachen.de

Aachener Informatik Bericht AIB-2015-03

Herausgeber: Fachgruppe Informatik
RWTH Aachen University
Ahornstr. 55
52074 Aachen
GERMANY

ISSN 0935-3232

Abstract

Model checking is an automated verification method guaranteeing that a mathematical model of a system satisfies a formally described property. It can be used to assess both qualitative and quantitative properties of complex software and hardware systems. Model checking suffers from the well-known *state space explosion* problem where the number of states grows exponentially in the number of program variables, channels and parallel components. Reduction techniques can be used to shrink the state space of system models by hiding redundant information and removing irrelevant details. The reduced state space can then be used for analysis provided it preserves a rich class of properties of interest. This thesis presents reduction techniques for a wide range of nondeterministic and probabilistic models. Our reduction techniques are based on the notions of *equivalence relations* and *layering*.

Equivalence relations reduce the state space of system models, by aggregating equivalent states into a single state. The reduced state space obtained under an equivalence relation, is called a quotient system. An example equivalence relation that is widely used to reduce the state space of nondeterministic and probabilistic models is *bisimulation*. On the other hand, layering involves carrying out structural transformations for the systems that are modeled as a network of system models, e.g., distributed systems. As a result of these structural transformations, the new state space obtained is smaller than the original non-layered one.

The first part of this thesis focuses on developing new equivalence relations for nondeterministic and Markovian models. For each of these relations, we define a quotient system, investigate its relationship with bisimulation and prove that it preserves interesting linear-time properties.

In the second part of this thesis we focus on layering based state space reduction for more expressive specification formalisms that support a stepwise refinement methodology. We develop a framework of layering for modal transition systems and probabilistic versions thereof. This involves a layered composition

operator, formulating *communication closed layer* (CCL) laws and defining a partial order (po) equivalence between modal transition systems. We also prove that the reduced model obtained as a result of applying layered transformation preserves reachability properties. As a result, reachability properties can be checked on the layered, typically smaller, model.

To summarize, this thesis presents the theoretical underpinnings of a number of novel reduction techniques for nondeterministic and probabilistic systems.

Zusammenfassung

Modelchecking ist ein automatisiertes Verfahren in der Verifikation, welches sicherstellt, dass ein mathematisches Modell eines Systems eine geforderte, formal definierte Eigenschaft besitzt. Es kann benutzt werden um sowohl qualitative als auch quantitative Eigenschaften komplexer Software- und Hardwaresysteme zu überprüfen. Beim Modelchecking stellt die *Zustandsraumexplosion* ein großes Problem dar. Dabei wächst die Anzahl der Zustände exponentiell mit der Anzahl der Programmvariablen, Kommunikationskanäle und Komponenten. Dem kann durch verschiedene Reduktionstechniken entgegengewirkt werden. Diese verringern den Zustandsraum, indem sie redundante Information verstecken oder irrelevante Details aus dem Model entfernen. Solange diese Manipulationen keinen Einfluss auf die Gültigkeit der zu überprüfenden Eigenschaften haben, können anschließend Analysetechniken auf dem verkleinerten Modell ausgeführt werden. Diese Arbeit beschäftigt sich mit Reduktionstechniken für ein breites Spektrum von nichtdeterministischen und probabilistischen Modellen. Die hier vorgestellten Techniken beruhen auf den Konzepten der *Äquivalenzrelationen* und *Layering* (engl. für "Schichtung").

Äquivalenzrelationen verkleinern den Zustandsraum, indem sie äquivalente Zustände zu einem einzigen zusammenfassen. Den auf dieser Weise reduzierten Zustandsraum nennt man Quotientenraum. Ein prominentes Beispiel einer Äquivalenzrelation, die zur Verkleinerung von nichtdeterministischen und probabilistischen Modellen benutzt wird, ist die *Bisimulation*. Auf der anderen Seite führt Layering strukturelle Transformationen auf dem gegebenen Modell durch. Das Ergebnis dieser strukturellen Transformationen ist ein kleinerer Zustandsraum als im Originalsystem. Dabei eignet sich Layering insbesondere für Systeme, die durch ein Netzwerk von Teilmodellen beschrieben werden, zum Beispiel verteilte Systeme.

Der erste Teil dieser Arbeit beschäftigt sich mit der Entwicklung neuer Äquivalenzrelationen für nichtdeterministische und Markow'sche Modelle. Für jede

solche Relation definieren wir ein Quotientensystem, untersuchen die Beziehung zur Bisimulationsrelation und zeigen, dass sie interessante Linearzeiteigenschaften erhält.

Im zweiten Teil konzentrieren wir uns auf Zustandsraumreduktion durch Layering auf ausdrucksstärkeren Modellen, die eine schrittweise Verfeinerungsmethodik erlauben. Wir entwickeln einen Ansatz um Layering auf modale Transitionssysteme und deren probabilistische Erweiterung anzuwenden. Dieser beinhaltet einen "geschichteten" Kompositionsoperator, der Regeln für eine *geschlossene Kommunikationsschicht* (communication closed layer (CCL)) vorgibt und eine partielle Ordnungsäquivalenz zwischen modalen Transitionssystemen definiert. Außerdem zeigen wir, dass Zustandsraumverkleinerung mit Hilfe von Layering Erreichbarkeitseigenschaften erhält. Folglich können Erreichbarkeitseigenschaften auf dem typischerweise kleineren Modell überprüft werden.

Insgesamt behandelt die vorliegende Arbeit die theoretischen Grundlagen für eine Reihe von neuen Reduktionstechniken für nichtdeterministische und probabilistische Systeme.

Acknowledgment

I would like to express my sincere gratitude to my PhD supervisor prof. dr. ir. Joost-Pieter Katoen for his continuous support, stimulating suggestions and guidance throughout the duration of my doctoral study. Professor Katoen's various proposals and reviews on my research work has always given me motivation and confidence in continuing my doctoral study. Without his encouragement and suggestions, this thesis might not have seen its completion.

I would like to thank the reading and defense committee: prof. dr. Holger Hermanns, prof. dr. Bernhard Rumpe, and prof. dr. Christof Löding. The valuable comments and feedback provided by them has significantly improved the material of this thesis.

I would also like to thank my (former) colleagues and friends of Software Modeling and Verification group for their support and help during this period: Souy, Friedrich, Sri, Falak, Sabrina, Alex, Stephen, Viet, Christian, Christina, Jonathan, Thomas, Arnd, Haidi, Erika, Xin, Hongfei, Harold, Weijie, Xiaoxiao, Nils, Elke and Birgit. A big thanks to Ian Larson for modeling the mutual exclusion case study and conducting the experiments.

Finally, I would like to thank my family, especially my father, my mother, my elder brother, bhabhi and chacha for their everlasting support, unconditional love and care.

0. ACKNOWLEDGMENT

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

Formal verification is the process of checking whether a design satisfies some requirements (properties). Model checking is the most popular method for automatic formal verification of safety critical software and hardware systems [9]. A model checker is a tool that allows formulating the model of the system (that needs to be verified) and the property (that needs to be checked against the model) using some precise mathematical language. Once the system and the property have been formally specified, it uses a model checking algorithm which carries out an exhaustive state space exploration to check whether the mathematical model satisfies the given property. In case the property is violated, a counterexample is generated for debugging the system. Model checking can be used to assess both qualitative and quantitative properties of complex systems.

In the qualitative setting, the system behavior is captured using Kripke structures (KSs), i.e., directed graphs where nodes represent labeled states of the system, and edges represent transitions. A property can be specified using temporal logics or by an automaton. Some example temporal logics that can be used to specify properties on KSs are linear temporal logic [131] (LTL), computation tree logic [35] (CTL) and CTL*. Similarly, deterministic Rabin automata (DRA) or Büchi automata [150] can be used to specify ω-regular properties. Tools such as the Simple Promela Interpreter (SPIN) [78], New Symbolic Model Verifier (NuSMV) [34], mCRL2 [69] and Construction and Analysis of Distributed Processes (CADP) [59] have been developed and successfully applied to the qualitative analysis of a diverse range of systems.

In the probabilistic context, Markovian models are typically used to capture

the behavior of systems that are subject to uncertainties. A discrete-time Markov chain (DTMC) is a Kripke structure in which each transition is equipped with a discrete probability describing the likelihood of moving from one state to another in a single move. In addition, in a continuous-time Markov chain (CTMC) state residence times are governed by negative exponential distributions. A probabilistic automaton (PA) can be used to capture the discrete-time probabilistic and nondeterministic behavior of systems. PAs are widely used for modeling randomized distributed algorithms and communication protocols [139]. Similarly, an interactive Markov chain (IMC) comprises both nondeterministic choices and exponentially distributed delays [74]. IMCs are compositional and have been used as semantic model for amongst others dynamic fault trees [23], architectural description languages such as AADL (Architectural Analysis and Design Language) [25], and generalised stochastic Petri nets. They are also used for stochastic extension of Statemate [21] and for hardware design [38, 39].

Quantitative properties can be specified on these models using probabilistic temporal logics or Büchi automata. In the discrete-time setting, LTL, probabilistic CTL (PCTL) [73], PCTL* and DRA can be used as property specification formalisms. For CTMCs, metric temporal logic (MTL) [95], deterministic timed automata (DTA) [1], continuous stochastic logic (CSL) [10] and CSL^{TA} [50] are example formalisms that can be used to specify real-time objectives. An example linear real-time objective that can be specified using DTA is: what is the probability to reach a given target state within the deadline, while avoiding "forbidden" states and not staying too long in any of the "dangerous" states on the way. Model checking tools such as Probabilistic Symbolic Model Checker (PRISM) [98], Markov Reward Model Checker (MRMC) [89] and PEPA Workbench [63] have been successfully applied for the modeling and analysis of probabilistic systems. Tools such as CADP [59] and the Interactive Markov Chain Analyzer (IMCA) [71] have been developed and used for the quantitative analysis of IMCs.

Besides KSs and its probabilistic variants, more expressive specification formalisms that support stepwise refinement methodology and compositionality have been proposed in the literature. These formalisms can be used for the stepwise design and analysis of complex systems. In this setting, a high level model of the system where the implementation details are hidden is constructed and used for the verification of interesting high level properties. More details can be added to the model with each refinement step and a final implementation can be obtained by applying a series of refinement steps.

For the qualitative case, modal transition systems (MTSs) [102, 106] extend

labeled transition systems [9, 113] (LTSs[1]) by providing support for partial specifications. MTSs are LTSs with two kinds of transitions, termed *may* and *must* transitions, satisfying the consistency condition that every must transition is also a may transition. An MTS can be refined by preserving at least all must transitions (and maybe adding some) while eliminating some may transitions. Model checking MTSs involves checking whether all the implementations satisfy a given temporal logic property or where there does exist an implementation that satisfies the property. MTSs have been successfully applied in program analysis [79, 138], model checking [28, 105], equation solving [107], interface theories [134, 153], component-based software development [133] and software product lines [70, 103].

In the probabilistic context, abstract probabilistic automata (APAs) [44, 47] have been recently defined as a complete specification and abstraction theory for PAs. The theory of APAs is equipped with parallel and conjunction operators, and allows comparing two APAs using a refinement relation. A satisfaction relation is used to check whether a PA is an implementation of a given APA. APA specifications can be used for the stepwise design and analysis of randomized distributed systems.

Model checkers both in classical and probabilistic setting, suffer from the state space explosion problem [9]: model checkers are often unable to explore completely any non-trivial logically bounded state space making it hard to provide any degree of assurance for reliability. This is primarily due to the large number of components running in parallel and data variables that are used while developing the model in a high level formal specification language. To combat this problem, various reduction techniques have been investigated. Reduction techniques involve collapsing sets of concrete states to abstract states and removing irrelevant details from the system model. The smaller model obtained after reduction can be used for analysis provided it still preserves the behavior that needs to be verified. Some example reduction techniques that can be used to reduce the state space of nondeterministic and probabilistic models are symmetry reduction [52, 99], bisimulation minimization [9, 10, 15], partial-order reduction [8, 9, 62, 128], predicate abstraction [67, 156] and three-valued abstraction [53, 92]. This thesis presents reduction techniques for nondeterministic and probabilistic models based on the notions of equivalence relations and layering.

Equivalence relations. Equivalence relations are used to compare the be-

[1]Note that LTSs are directed graphs where transitions are labeled with action names. KSs and LTSs are equally expressive and one can find several embeddings of one of these models into the other [42, 43, 136].

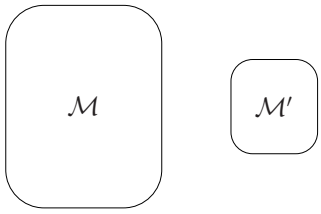

Figure 1.1: State space reduction under an equivalence relation $\mathcal{R}$

havior of two models and reduce the state space of a system model by combining equivalent states into a single state. The reduced state space obtained under an equivalence relation, called a quotient system, can then be used for model checking provided it preserves a rich class of properties of interest. The main principle is captured in Fig. 1.1 where quotient $\mathcal{M}'$ is obtained from model $\mathcal{M}$ under an equivalence relation $\mathcal{R}$. For nondeterministic and probabilistic models, one usually distinguishes between linear-time and branching-time equivalence relations [9, 10, 16, 64, 157]. Trace equivalence is one of the most widely used equivalence relations to compare the linear-time behavior of models. For KSs, two states are trace equivalent if the possible sequences of words starting from these states are the same [76, 135, 148]. Additionally, for Markov chains, the possible sequences of words need to have the same probability [16, 157]. Several extensions of trace equivalences have been proposed for KSs and Markov chains, e.g., testing, failure and readiness semantics [15, 16, 157]. Similarly, in the weak setting (where stutter steps are allowed), stutter trace equivalence has been proposed where a pair of sequences is considered to be equivalent if they differ in at most the number of times a set of propositions may adjacently repeat.

In the branching-time setting, bisimulation relations [9, 10] are one of the most important reduction techniques based on equivalence relations that can be used to substantially reduce the state-space of models to be verified. Bisimulation minimization preserves linear-time (LT) and branching-time (BT) properties [9, 10]. The condition to exhibit identical stepwise behavior is slightly relaxed in case of simulation relations [9, 10]. Stuttering variants of bisimulation and simulation pre-orders have also been defined for KSs and Markov models [9, 10]. Several papers report data showing that bisimulation minimization and use of simulation relations can substantially reduce the state-space of models to be verified [5, 55, 91]. Unfortunately, bisimulation is too restrictive as it requires equivalent states to simulate their mutual stepwise behavior, and it is often desirable to obtain a quotient system smaller than bisimulation such that properties of interest are still

preserved. This is particularly important if the properties to be verified belong to the class of linear-time properties, e.g., safety properties, liveness properties and in general ω-regular properties. These properties can be expressed using LTL, Property Specification Language (PSL) [81], DRA, Büchi automata and DTA/MTL (for linear real-time objectives). An example class of systems where bisimulation usually fails to provide any state space reduction is incremental service systems [17] (Section 5.2).

This thesis focuses on an equivalence relation that allows for a more aggressive state space reduction than bisimulation. Roughly speaking, two states s, s' are related under this new equivalence if each pair of direct predecessors of C s.t. $s, s' \in C$, moves to the same equivalence class in two steps via C. For Markov chains, the *weighted* probability (rate) of doing so should coincide. Note that we use a two-step perspective for combining multiple states into a single state. This merging strategy is different from that of bisimulation which requires two states s, s' to exhibit identical stepwise behavior. This new equivalence relation can be seen as a state space reduction technique induced by trace/testing equivalence [16] (for KSs and Markov chains).

Layering. Layering is useful for the state space reduction of models capturing the behavior of distributed systems [82]. Model construction in these systems involves composing several components in parallel, where each component usually has multiple sub-components that are executed in a sequential manner. Components cooperate through their synchronization over common actions and through their respective action dependencies. Action dependencies between sub-components can be either explicitly stated or derived from the operations performed on data variables that are updated during an action execution (in case of systems with data variables). Some example systems that have this structure are distributed algorithms such as the randomized mutual exclusion algorithm by Kushilevitz and Rabin [96], Fischer's real-time mutual exclusion protocol [85], the two phase commit protocol [20], and the distributed minimum weight spanning tree algorithm [58].

The main principle of layering based reduction is illustrated in Fig. 1.2. Here two components $\mathcal{M}$ and $\mathcal{N}$ are composed in parallel (left), where each component consists of n sub-components which are executed in a sequential manner (denoted by ;). The system obtained after performing layered transformation is shown in Fig. 1.2 (right). All the sub-components of $\mathcal{M}$ and $\mathcal{N}$ are assumed to be acyclic, and can be repeated by allowing top-level recursion in $\mathcal{M}$ and $\mathcal{N}$ (as indicated by *). In other words, every component ($\mathcal{M}$ and $\mathcal{N}$) can have multiple rounds

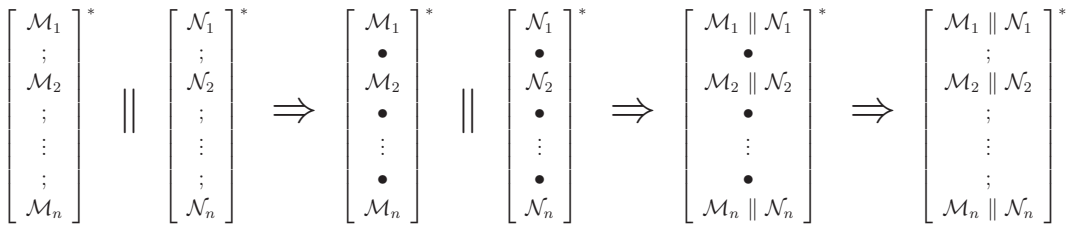

Figure 1.2: Layered reduction

	Parallel	Layered
Build time (s)	898.70	90.39
# States	198063	71619
# Transitions	351432	128920

Table 1.1: Parallel vs. layered composition

of execution, where a new round is started only when the last sub-component of the previous round, i.e., $\mathcal{M}_n$ has been completed. This is important as deadlocks are usually considered to be undesirable for distributed algorithms.

Roughly speaking, layering exploits the independence between sub-components to transform the system under consideration from a distributed representation to a layered representation. These transformations are syntactic: the idea is to apply a series of transformations to model descriptions, yielding a layered representation (cf. Fig. 1.2, right). For the intermediate transformations, a layered composition operator "$\bullet$" is used to denote the layered representation of the system. Informally, $\mathcal{M} \bullet \mathcal{N}$ allows synchronization on common actions and interleaving on disjoint actions, except when some action a of $\mathcal{N}$ depends on one or more actions of $\mathcal{M}$; in this case, a can be executed only after all the actions of $\mathcal{M}$ on which it depends have been executed. This new composition operator allows formulating *Communication Closed Layer* (CCL) laws [82], which are required to carry out the structural transformations and establish an equivalence between the two systems. Since the sub-components within a component are executed sequentially, a partial order relation is used to relate the $\bullet$ and ; (sequential composition) operator. The amount of state space reduction that can be achieved using layering is indicated in Table 1.1. Here the results of our implementation[1] for the layered analysis of a randomized mutual exclusion algorithm [149] have

[1]This case study was modeled using the PRISM model checker [98].

been presented for 3 processes and 5 rounds. These results clearly indicate that layered reasoning can significantly reduce the state space of system models capturing the behavior of distributed systems. We propose a framework of layered reduction for cyclic sequential composition of acyclic MTSs and APAs.

1.2 Outline of the Thesis

- **Chapter 2** recalls the basic concepts of nondeterministic and probabilistic models that form the basis of this thesis.

- **Chapter 3** proposes Kripke minimization equivalence (KME) and weak Kripke minimization equivalence (WKME) for KSs. We show that KMEs and WKMEs can be used for repeated minimization of a KS and union of KMEs (resp. WKMEs) is not necessarily a KME (resp. WKME). We define the quotient system under these relations and investigate the relationship between the new relations and strong bisimulation and divergence-sensitive stutter bisimulation, respectively. Next, we prove that linear-time (LT) properties and stutter-insensitive LT properties are preserved under KME and WKME quotienting, respectively. We also prove that KME is a congruence w.r.t. synchronous parallel composition. The results presented in this chapter are based on the following work:

  Arpit Sharma. *A Two Step Perspective for Kripke Structure Reduction*. In 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Student Research Forum. pages 29–41. ISBN 978-80-87136-15-7, 2013.

- **Chapter 4** proposes weighted probabilistic equivalence (WPE) for DTMCs. We show that WPEs can be used for repeated minimization of a DTMC and union of WPEs is not necessarily a WPE. We define the quotient system under WPE and investigate its relationship with probabilistic bisimulation for DTMCs. Next, we prove that the probability of satisfying ω-regular properties is preserved under WPE quotienting. We also show that WPE is a congruence w.r.t. synchronous parallel composition for DTMCs. Finally, we extend these preservation results to DTMCs with rewards. The results presented in this chapter are based on the following work:

  Arpit Sharma. *Weighted Probabilistic Equivalence Preserves ω-Regular Properties*. In 16th Measurement, Modeling, and Evaluation of Computing

Systems and Dependability and Fault Tolerance Conference (MMB/DFT). pages 121–135. Volume 7201 of LNCS. Springer Verlag, 2012.

- **Chapter 5** proposes weighted lumpability (WL) for CTMCs. We show that WL relations can be used for repeated minimization of a CTMC and union of WL relations is not necessarily a WL. We define the quotient system under WL and investigate its relationship with stochastic bisimulation for CTMCs. Next, we prove that the probability of satisfying a deterministic timed automaton (DTA) is preserved under WL quotienting. Finally, we show that the probability of satisfying a metric temporal logic (MTL) formula is also preserved under WL quotienting. The results presented in this chapter are based on the following work:

  Arpit Sharma, and Joost-Pieter Katoen. *Weighted Lumpability on Markov Chains*. In 8th Ershov Informatics Conference (PSI). pages 322–339. Volume 7162 of LNCS. Springer Verlag, 2012.

- **Chapter 6** proposes a state space reduction technique for a network of MTSs based on layered composition. We formulate communication closed layer (CCL) laws and define a partial order equivalence (po) between MTSs. Next, we show that layered and sequential composition are po-equivalent and satisfy the same existential ($\exists$) and universal ($\forall$) reachability properties. The results presented in this chapter are based on the following work:

  Arpit Sharma, and Joost-Pieter Katoen. *Layered Reduction for Modal Specification Theories*. In 10th International Symposium on Formal Aspects of Component Software (FACS). Volume 8348 of LNCS. Springer, 2013.

- **Chapter 7** proposes a state space reduction technique for a network of APAs based on layered composition. We formulate communication closed layer (CCL) laws and define a partial order (po) equivalence between APAs. Next, we show that layered and sequential composition are po-equivalent and have the same extremal probabilities to reach the set of final states. The results presented in this chapter are based on the following work:

  Arpit Sharma, and Joost-Pieter Katoen. *Layered Reduction for Abstract Probabilistic Automata*. In 14th International Conference on Application of Concurrency to System Design (ACSD). IEEE, 2014.

- **Chapter 8** proposes interactive Markovian equivalence (IME) and weak interactive Markovian equivalence (WIME) for closed IMCs. Next, we show

that IMEs and WIMEs can be used for repeated minimization of a closed IMC and union of IMEs (resp. WIMEs) is not necessarily an IME (resp. WIME). We define the quotient system under these relations and investigate the relationship between the new relations and bisimulation and weak bisimulation for closed IMCs, respectively. We also show that the theory of layering cannot be extended to IMCs, which can be used for analysis of distributed algorithms with random times. More specifically, we show that it is not possible to relate the sequential and layered composition operator by defining a po equivalence or a simulation preorder, such that linear real-time properties are preserved. The results presented in this chapter are new and not published.

- **Chapter 9** concludes the thesis and presents some directions for future research.

All the proofs are contained in the Appendix A.

Chapter 2

Preliminaries

2.1 Nondeterministic Models

Kripke structures (KSs) and labeled transition systems (LTSs) are convenient formalisms for modeling and analysis of system behavior. KSs are state-based models (states are labeled with atomic propositions) and LTSs are event-based models (transitions are labeled with action names). KSs are commonly used for model checking temporal logic formulas (e.g., LTL [9, 131] and CTL [9, 35]) that specify the desired qualitative behavior of systems. On the other hand, LTSs are often used as a semantic model for process algebraic languages, e.g., communicating sequential processes (CSP) [77] and calculus of communicating systems (CCS) [110]. Both these models are equally expressive and several embeddings have been proposed in the literature which show that KSs and LTSs are interchangeable w.r.t. equivalences and temporal logics [42, 43, 136]. In this section we present the basic concepts of KSs and LTSs.

2.1.1 Kripke Structures

Definition 2.1 (KS) *A Kripke structure (KS) is a tuple* $\mathcal{K} = (S, \rightarrow, AP, L, s_0)$ *where:*

- S *is a non-empty finite set of states,*
- $\rightarrow \subseteq S \times S$*, is a transition relation s.t.* $\forall s \in S \exists s' \in S$ *with* $(s, s') \in \rightarrow$*,*
- AP *is a finite set of atomic propositions,*
- $L : S \rightarrow 2^{AP}$ *is a labeling function,*

- $s_0 \in S$ *is the initial state.*

For simplicity, we write $s \to s'$ instead of $(s, s') \in \to$. Let $s \in S$ and $C \subseteq S$, then $Post(s, C) = \{s' \in C \mid s \to s'\}$. Let $Post(s) = \{s' \in S \mid s \to s'\}$. For $C \subseteq S$, let $Pred(C) = \{s' \in S \mid \exists s \in C.s' \to s\}$.

Definition 2.2 (KS paths) *Let $\mathcal{K} = (S, \to, AP, L, s_0)$ be a KS. An infinite* path *π in $\mathcal{K}$ is an infinite state sequence, i.e., $s_0 \to s_1 \to s_2 \ldots \in S^\omega$ with $s_i \in S$.*

Note that, since we do not allow KS $\mathcal{K}$ to have terminal states, i.e., which do not have any outgoing transitions, we only consider infinite paths (starting from the initial state). Let $Paths^{\mathcal{K}}(s_0)$ denote the set of all infinite paths in $\mathcal{K}$ that start in s_0. For infinite path π and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i+1)$-st state of π. Let $\pi[i...]$ denote the suffix of path π starting in the $(i+1)$-st state.

Definition 2.3 (KS traces) *Let $\mathcal{K} = (S, \to, AP, L, s_0)$ be a KS. The* trace *of an infinite path $\pi = s_0 \to s_1 \to s_2 \ldots \in S^\omega$ is $trace(\pi) = L(s_0)L(s_1)L(s_2)\ldots \in (2^{AP})^\omega$.*

Intuitively a trace of an infinite path is the infinite sequence of sets of atomic propositions that are valid in the states of the path, i.e., a trace is an infinite word over the alphabet 2^{AP}. Let $Traces^{\mathcal{K}}(s_0)$ denote the set of all infinite traces in $\mathcal{K}$ that start in s_0.

Example 2.4 *Consider the KS $\mathcal{K}$ shown in Fig. 2.1, where we have $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, $AP = \{a, b\}$ and s_0 is the initial state. An example infinite path π of $\mathcal{K}$ is $s_0 \to s_1 \to s_4 \to s_6 \ldots$. Here we have $\pi[3] = s_6$. The trace for path π is given by $trace(\pi) = \{a\}\varnothing\{a\}\{b\}\ldots$.*

Definition 2.5 (Trace-equivalence) *Let $\mathcal{K} = (S, \to, AP, L, s_0)$ be a KS and $\pi_i \in Paths^{\mathcal{K}}(s_0)$, $i = 1, 2$. π_1 and π_2 are* trace-equivalent*, denoted by $\pi_1 \triangle \pi_2$, if $L(\pi_1[i]) = L(\pi_2[i])$ for all $i \geq 0$.*

Definition 2.6 (Stutter step) *Transition $s \to s'$ in KS $\mathcal{K} = (S, \to, AP, L, s_0)$ is a* stutter step *if $L(s) = L(s')$.*

The notion of stuttering is lifted to paths as follows.

Definition 2.7 (Stutter-equivalent paths) *Let $\mathcal{K} = (S, \to, AP, L, s_0)$ be a KS and $\pi_i \in Paths^{\mathcal{K}}(s_0)$, $i = 1, 2$. π_1 and π_2 are* stutter-equivalent*, denoted*

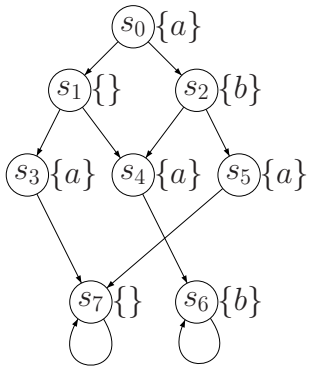

Figure 2.1: An example KS $\mathcal{K}$

by $\pi_1 \triangleq \pi_2$*, if there exists an infinite sequence* $A_0A_1A_2\ldots$ *with* $A_i \subseteq AP$ *and natural numbers* $n_0, n_1, n_2, \ldots, m_0, m_1, m_2, \ldots \geq 1$ *s.t.*

$$trace(\pi_1) = \underbrace{A_0 \ldots A_0}_{n_0-times}\underbrace{A_1 \ldots A_1}_{n_1-times}\underbrace{A_2 \ldots A_2}_{n_2-times}\ldots$$

$$trace(\pi_2) = \underbrace{A_0 \ldots A_0}_{m_0-times}\underbrace{A_1 \ldots A_1}_{m_1-times}\underbrace{A_2 \ldots A_2}_{m_2-times}\ldots$$

where $\underbrace{A_0 \ldots A_0}_{n_0-times}$ *denotes for all* $i = 0 \ldots n_0 - 1$*,* $L(\pi_1[i]) = A_0$*.*

Note that $\underbrace{A_0 \ldots A_0}_{n_0-times}$ only refers to the first block, for other blocks it is defined in an analogous manner. Accordingly, stutter-equivalence for any two infinite traces $\rho_1, \rho_2 \in (2^{AP})^\omega$ (denoted by $\rho_1 \triangleq \rho_2$) can be defined.

Assumptions. Throughout this thesis we assume that every state of KS $\mathcal{K}$ has at least one predecessor, i.e., $Pred(s) = \{s' \in S \mid s' \to s\} \neq \varnothing$ for any $s \in S$. This is not a restriction, as any KS $(S, \to, AP, L, s_0)$ can be transformed into an equivalent KS $(S', \to', AP', L', s_0')$ which fulfills this condition. This is done by adding a new state $\hat{s}$ to S equipped with a self-loop and which has a transition to each state in S without predecessors. To distinguish this state from the others we set $L'(\hat{s}) = \bot$ with $\bot \notin AP$. (All other labels, states and transitions remain unaffected.) Let $s_0' = s_0$. It follows that all states in $S' = S \cup \{\hat{s}\}$ have at least one predecessor. Moreover, the reachable state space of both KSs coincides. We also assume that the initial state s_0 of a KS is distinguished from all other states by a unique label, say \$. This assumption implies that for any equivalence

that groups equally labeled states, $\{s_0\}$ constitutes a separate equivalence class. Both assumptions do not affect the basic properties of the KS such as linear or branching-time properties. For convenience, we neither show the state $\hat{s}$ nor the label \$ in figures. This assumption is required as Chapter 3 proposes an equivalence relation for KSs that checks reachability from predecessors of every equivalence class to its successor equivalence classes.

2.1.2 Labeled Transition Systems

Definition 2.8 (LTS) *A labeled transition system (LTS) is a tuple* $\mathcal{T} = (S, Act, s_0, S_f, V)$ *where:*

- S *and* s_0 *are defined as before,*
- Act *is a finite set of actions,*
- $S_f \subset S$ *is the set of final states where* $s_0 \notin S_f$*,*
- $V : S \setminus S_f \times Act \times S \to \mathbb{B}_2$ *is a two-valued transition function.*

Here $\mathbb{B}_2 = \{\bot, \top\}$, with $\bot < \top$. $V(s, a, s')$ identifies the a-labeled transition of the automaton in state s: $\top$ indicates its presence and $\bot$ indicates its absence. We write $s \xrightarrow{a} s'$ if $V(s, a, s') = \top$. Labeled transition systems are basically directed graphs where nodes represent states, and edges model transitions, i.e., state changes. Transitions specify how the system can evolve from one state to another. In case a state has more than one outgoing transition, the next transition is chosen in a purely non-deterministic fashion. A possible behaviour in an LTS is obtained from the resolution of non-deterministic choices, described in terms of paths. A path π of LTS $\mathcal{T}$ is a (possibly infinite) sequence of the form $\pi = s_0 a_1 s_1 a_2 s_2 a_3 \ldots$ where $\forall n : s_n \xrightarrow{a_{n+1}} s_{n+1}$. Let $last(\pi)$ denote the last state of π (if π is finite). Let $|\pi|$ be the length (number of actions) of a finite path π. For infinite path π and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i+1)$-st state of π. For finite path π of length n, $\pi[i]$ is only defined for $i \leq n$ and defined as for infinite paths. Let $Paths_{fin}(\mathcal{T})$ be the set of all finite paths in LTS $\mathcal{T}$, and $Paths_{inf}(\mathcal{T})$ the set of all infinite paths of $\mathcal{T}$ that start in state s_0. Let $Paths_{fin}^{S_f}(\mathcal{T})$ be the set of all finite paths of $\mathcal{T}$ that start in state s_0 and end in some state $s \in S_f$.

Definition 2.9 (Deterministic LTS) *LTS* $\mathcal{T} = (S, Act, s_0, S_f, V)$ *is deterministic, if for every state* s *and action* a *we have:* $|\{s' \in S \mid V(s, a, s') \neq \bot\}| \leq 1$.

In simple words, an LTS $\mathcal{T}$ is deterministic if none of its states has multiple outgoing transitions labeled with the same action.

2.2 Stochastic Models

Markov chains [9, 93] have a wide applicability ranging from classical performance and dependability evaluation to systems biology. A discrete-time Markov chain (DTMC) is a Kripke structure in which each transition is equipped with a discrete probability describing the likelihood of moving from one state to another in a single move. In addition, in a continuous-time Markov chain (CTMC) state residence times are governed by negative exponential distributions. We start this section by recalling the basic concepts of DTMCs and CTMCs. Models that extend Markov chains with support for nondeterminism will be discussed in Section 2.2.3 and Section 2.2.4.

2.2.1 Discrete-Time Markov Chains

Definition 2.10 (DTMC) *A (labeled)* discrete-time Markov chain *(DTMC) is a tuple* $\mathcal{D} = (S, P, AP, L, s_0)$ *where:*

- S, AP, L *and* s_0 *are defined as before,*
- $P : S \times S \to [0, 1]$ *is a probability matrix such that* $\sum_{s' \in S} P(s, s') = 1$ *for all* $s \in S$.

Intuitively, $P(s, s')$ specifies the probability to move from state s to s' in one step, i.e., by a single transition. State s of DTMC $\mathcal{D}$ is called absorbing if and only if $P(s, s) = 1$, and $P(s, s') = 0$ for all $s' \in S$ s.t. $s \neq s'$.

Definition 2.11 (DTMC paths) *Let* $\mathcal{D} = (S, P, AP, L, s_0)$ *be a DTMC. An infinite* path π *in* $\mathcal{D}$ *is an infinite state sequence* $s_i \in S$, *i.e.,* $s_0 \to s_1 \to s_2 \ldots \in S^\omega$ *such that* $P(s_i, s_{i+1}) > 0$, *for all* $i \geq 0$. *A finite path* π *is a finite prefix of an infinite path.*

For path $\pi \in \mathcal{D}$, $inf(\pi)$ denotes the set of states that are visited infinitely often in π. For finite DTMCs, $inf(\pi)$ is nonempty for all infinite paths π. Let $Paths^{\mathcal{D}} = Paths^{\mathcal{D}}_{fin} \cup Paths^{\mathcal{D}}_{\omega}$ denote the set of all paths in $\mathcal{D}$, where $Paths^{\mathcal{D}}_{fin} = \bigcup_{n \in \mathbb{N}} Paths^{\mathcal{D}}_{n}$ is the set of all finite paths in $\mathcal{D}$ and $Paths^{\mathcal{D}}_{\omega}$ is the set of all infinite paths in $\mathcal{D}$. For infinite path π and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i + 1)$-st state of π. For finite path π, which is a finite prefix of length n of an infinite path, $\pi[i]$ is only defined for $i \leq n$ and defined as in the case of infinite paths. Let $Paths(s_0)$ denote the set of all paths that start in s_0. Let $\pi[i...]$ denote the suffix of path π starting in the $(i + 1)$-st state. Let $Pred(s) = \{s' \in S \mid P(s', s) > 0\}$ and $Pred(C) = \bigcup_{s \in C} Pred(s)$ for $C \subseteq S$.

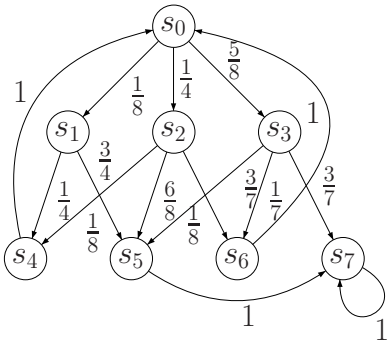

Figure 2.2: An example DTMC $\mathcal{D}$

Example 2.12 *Consider the DTMC $\mathcal{D}$ shown in Fig. 2.2, where we have $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, $AP = \{a, b, c\}$, $L(s_0) = \{c\}$, $L(s_7) = \{\}$, $L(s_1) = L(s_2) = L(s_3) = \{b\}$, $L(s_4) = L(s_5) = L(s_6) = \{a\}$ and s_0 is the initial state. The transition probabilities are attached to the transitions. An example finite path π of $\mathcal{D}$ is $s_0 \to s_1 \to s_5 \to s_7$. Here $\pi[2] = s_5$.*

Definition 2.13 (Cylinder set) *Let $s_0, \ldots, s_k \in S$ with $P(s_i, s_{i+1}) > 0$ for $0 \leq i < k$. $Cyl(s_0, \ldots, s_k)$ denote the* cylinder set *consisting of all paths $\pi \in Paths(s_0)$ such that $\pi[i] = s_i$ for $i \leq k$.*

Intuitively the cylinder set spanned by the finite path π consists of all infinite paths that start with π. The definition of a Borel space on paths of a DTMC follows [3, 54]. Let $\mathcal{F}(Paths(s_0))$ be the smallest σ-algebra on $Paths(s_0)$ which contains all sets $Cyl(s_0, \ldots, s_k)$ s.t. $s_0, \ldots, s_k$ is a state sequence with $P(s_i, s_{i+1}) > 0$, $(0 \leq i < k)$.

Definition 2.14 *The probability measure $\Pr_{s_0}$ on $\mathcal{F}(Path(s_0))$ is the unique measure defined by induction on k in the following way. Let $\Pr_{s_0}(Cyl(s_0)) = 1$ and for $k > 0$:*

$$\Pr_{s_0}(Cyl(s_0, \ldots, s_k, s')) = \Pr_{s_0}(Cyl(s_0, \ldots, s_k)) \cdot P(s_k, s')$$

For $T \subseteq S$ and $s \in S$, let $P(s, T) = \sum_{s' \in T} P(s, s')$ be the cumulative probability to directly move from state s to some state in $T \subseteq S$.

Definition 2.15 (SCC) *A subset T of S is called* strongly connected *if for each pair (s, t) of states in T there exists a path fragment $s_0 \to s_1 \ldots s_n$ such that $s_i \in T$ for $0 \leq i \leq n, s_0 = s$ and $s_n = t$. A* strongly connected component *(SCC) of $\mathcal{D}$ denotes a strongly connected set of states such that no proper superset of T is strongly connected.*

Definition 2.16 (BSCC) *A* bottom *strongly connected component (BSCC, for short) of $\mathcal{D}$ is an SCC T from which no state outside T is reachable, i.e., for each state $t \in T$ it holds that $P(t, T) = 1$.*

Let $BSCC(\mathcal{D})$ denote the set of all BSCCs of $\mathcal{D}$.

Theorem 2.17 *[9, pp. 775-776] For each state s of a finite DTMC $\mathcal{D}$:*

$$Pr_s\{\pi \in Paths(s) | inf(\pi) \in BSCC(\mathcal{D})\} = 1.$$

In simple words this theorem states that almost surely any finite DTMC eventually reaches a BSCC and visits all states of the BSCC infinitely often.

Example 2.18 *Consider the DTMC $\mathcal{D}$ in Fig. 2.2, the only BSCC in $\mathcal{D}$ is $\{s_7\}$. According to the previous theorem, any infinite path will almost surely lead to this BSCC.*

Assumptions. Like for KSs, we assume that every state of DTMC $\mathcal{D}$ has at least one predecessor and s_0 is distinguished from all other states by a unique label, say \$. Both assumptions do not affect the basic properties of the DTMC such as transient or steady-state distributions. This assumption is required as Chapter 4 proposes an equivalence relation for DTMCs that checks probabilistic reachability from predecessors of every equivalence class to its successor equivalence classes.

2.2.2 Continuous-Time Markov Chains

Definition 2.19 (CTMC) *A (labeled)* continuous-time Markov chain *(CTMC) is a tuple $\mathcal{C} = (S, R, AP, L, s_0)$ where:*

- *S, AP, L and s_0 are defined as before,*
- *$R : S \times S \to \mathbb{R}_{\geq 0}$ is a rate function,*

The *exit rate* $E(s)$ for state $s \in S$ is defined by $E(s) = \sum_{s' \in S} R(s, s')$. We assume that $\forall s \in S : E(s) \neq 0$. The semantics of a CTMC is defined as follows. The probability of moving from s to s' in a single step is defined by $P(s, s') = \frac{R(s,s')}{E(s)}$. The probability to exit state s within t time units is given by $1 - e^{-E(s) \cdot t}$. The probability to move from state s to s' within t time units equals $P(s, s') \cdot (1 - e^{-E(s) \cdot t})$.

Definition 2.20 (CTMC timed paths) *Let $\mathcal{C} = (S, R, AP, L, s_0)$ be a CTMC. An infinite* path *π in $\mathcal{C}$ is an alternating sequence of states $s_i \in S$ and time instants $t_i \in \mathbb{R}_{>0}$, i.e., $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \cdots$ such that $R(s_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. A finite path π is a finite prefix of an infinite path.*

Let $Paths^{\mathcal{C}} = Paths^{\mathcal{C}}_{fin} \cup Paths^{\mathcal{C}}_{\omega}$ denote the set of all paths in $\mathcal{C}$, where $Paths^{\mathcal{C}}_{fin} = \bigcup_{n\in\mathbb{N}} Paths^{\mathcal{C}}_{n}$ is the set of all finite paths in $\mathcal{C}$ and $Paths^{\mathcal{C}}_{\omega}$ is the set of all infinite paths in $\mathcal{C}$. For infinite path $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \cdots$ and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i+1)$st state of π. Let $\delta(\pi, i) = t_i$ be the time spent in state s_i. For any $t \in \mathbb{R}_{\geq 0}$ and i, the smallest index s.t. $t \leq \sum_{j=0}^{i} t_j$, let $\pi@t = \pi[i]$, the state occupied at time t. For finite path $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n$, which is a finite prefix of an infinite path, $\pi[i]$, $\delta(\pi, i)$ are only defined for $i \leq n$, and for $i < n$ defined as in the case of infinite paths. For all $t > \sum_{j=0}^{n-1} t_j$, let $\pi@t = s_n$; otherwise $\pi@t$ is defined as in the case of infinite paths. Let $\delta(\pi, n) = \infty$. Let $\alpha : S \to [0, 1]$, be the intial probability distribution s.t. $\sum_{s\in S} \alpha(s) = 1$. Since $\mathcal{C}$ has a single initial state s_0, $\alpha(s_0) = 1$, and $\forall s \in S$ s.t. $s \neq s_0$, $\alpha(s) = 0$. Let $Paths(s_0)$ denote the set of all paths that start in s_0.

Example 2.21 *Consider the CTMC $\mathcal{C}$ shown in Fig. 2.3, where $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, $AP = \{a, b\}$ and s_0 is the initial state. The transition rates are associated with the transitions. An example timed path π of $\mathcal{C}$ is $s_0 \xrightarrow{1.3} s_1 \xrightarrow{1.5} s_3 \xrightarrow{2} s_6$. Here we have $\pi[3] = s_6$ and $\pi@3 = s_3$.*

Definition 2.22 (Cylinder set) *Let $s_0, \ldots, s_k \in S$ with $P(s_i, s_{i+1}) > 0$ for $0 \leq i < k$ and $I_0, \ldots, I_{k-1}$ be nonempty intervals in $\mathbb{R}_{\geq 0}$. $Cyl(s_0, I_0, \ldots, I_{k-1}, s_k)$ denotes the* cylinder set *consisting of all paths $\pi \in Paths(s_0)$ s.t. $\pi[i] = s_i$ for $i \leq k$, and $\delta(\pi, i) \in I_i$ for $(i < k)$.*

The definition of a Borel space on paths of a CTMC follows [11]. Let $\mathcal{F}(Paths(s_0))$ be the smallest σ-algebra on $Paths(s_0)$ which contains all sets $Cyl(s_0, I_0, \ldots, I_{k-1}s_k)$ s.t. $s_0, \ldots, s_k$ is a state sequence with $P(s_i, s_{i+1}) > 0$ $(0 \leq i < k)$ and $I_0, \ldots, I_{k-1}$ ranges over all sequences of nonempty intervals in $\mathbb{R}_{\geq 0}$.

Definition 2.23 *The probability measure $\Pr_\alpha$ on $\mathcal{F}(Path(s_0))$ is the unique measure defined by induction on k in the following way. Let $\Pr_\alpha(Cyl(s_0)) = \alpha(s_0)$ and for $k > 0$:*

$$\Pr_\alpha(Cyl(s_0, I_0, \ldots, s_k, I', s')) = \Pr_\alpha(Cyl(s_0, I_0, \ldots, s_k)) \cdot P(s_k, s', I')$$

where $P(s_k, s', I') = P(s_k, s') \cdot \left(e^{E(s_k)\cdot a} - e^{E(s_k)\cdot b}\right)$ with $a = \inf I'$ and $b = \sup I'$.

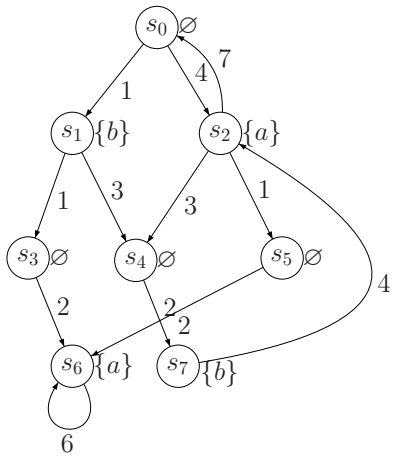

Figure 2.3: An example CTMC $\mathcal{C}$

Assumptions. Like for KSs and DTMCs, we assume that every state of CTMC $\mathcal{C}$ has at least one predecessor and s_0 is distinguished from all other states by a unique label, say \$. Both assumptions do not affect the basic properties of the CTMC such as transient or steady-state distributions. This assumption is required as Chapter 5 proposes an equivalence relation for CTMCs that checks weighted rates from predecessors of every equivalence class to its successor equivalence classes.

2.2.3 Probabilistic Automata

A probabilistic automaton (PA) resembles a labeled transition system (LTS) [113], but its transitions target probability distributions over states instead of single states. PAs have been developed by Segala [139, 141] and are compositional–a parallel composition operator allows one to construct a complex PA from several component PAs running in parallel, thus allowing to model complex systems in a modular way. PAs are widely used for the modeling and verification of randomized distributed algorithms and networking protocols. They have been used as semantic model for amongst others probabilistic process algebras [127] and the PIOA language [32]. Tools such as PRISM [98] have been successfully applied to model check quantitative properties on PAs.

Let S be a countable set. The function $\mu : S \to [0,1]$ is a *distribution* on S if $\sum_{s \in S} \mu(s) = 1$. Let $Dist(S)$ denote the set of distributions on S and $supp(\mu) = \{s \in S | \mu(s) > 0\}$ be the support of μ.

Definition 2.24 (PA) *A probabilistic automaton (PA) is a tuple* (S, s_0, S_f, Act, V)*, where:*

- S, s_0, S_f *and* Act *are defined as before,*
- $V : S \setminus S_f \times Act \times Dist(S) \to \mathbb{B}_2$ *is a two-valued transition function.*

Here $\mathbb{B}_2 = \{\bot, \top\}$, with $\bot < \top$. $V(s, a, \mu)$ identifies the transition of the automaton in state s: $\top$ indicates its presence and $\bot$ indicates its absence. We write $s \xrightarrow{a} \mu$ meaning $V(s, a, \mu) = \top$. Intuitively, PAs are very similar to LTSs, with the only difference that the target of each transition is a distribution over states instead of just a single state. Let $act(s)$ denote the set of enabled actions from state s, i.e., $act(s) = \{a \in Act \mid \exists \mu : V(s, a, \mu) \neq \bot\}$. PA $\mathcal{P}$ is *deterministic*, if for every state s and action a we have: $|\{\mu \in Dist(S) \mid V(s, a, \mu) \neq \bot\}| \leq 1$. In simple words, a PA $\mathcal{P}$ is deterministic if none of its states have multiple transitions on same action name. A possible behaviour of a PA is obtained from the resolution of non-deterministic and probabilistic choices, described in terms of paths. A path π of PA $\mathcal{P}$ is a (possibly infinite) sequence of the form $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 \ldots$ where $\forall n : s_n \xrightarrow{a_{n+1}} \mu_{n+1}$, and $\mu_{n+1}(s_{n+1}) > 0$. Let $last(\pi)$ denote the last state of π (if π is finite). Let $|\pi|$ be the length (number of actions) of a finite path π. For infinite path π and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i+1)$-st state of π. For finite path π of length n, $\pi[i]$ is only defined for $i \leq n$ and defined as for infinite paths. Let $Paths_{fin}(\mathcal{P})$ be the set of all finite paths in PA $\mathcal{P}$, and $Paths_{inf}(\mathcal{P})$ the set of all infinite paths of $\mathcal{P}$ that start in state s_0. Let $Paths_{fin}^{S_f}(\mathcal{P})$ be the set of all finite paths of $\mathcal{P}$ that start in state s_0 and end in some state $s \in S_f$. A trace of a finite path π is the sequence of actions obtained by removing the states (and the distributions).

Example 2.25 *Consider the PA $\mathcal{P}$ in Fig. 2.4 (left), where $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$, $Act = \{c, w, r, h, t\}$, s_0 is the initial state, and $S_f = \{s_7, s_8\}$. Here s_0 can move with action c to s_1 and s_2 with probability 0.5 and 0.5, respectively. An example finite path π is $s_0 c \mu_1 s_1 w \mu_2 s_3 r \mu_3 s_5 h \mu_4 s_7$, where $\mu_1(s_1) = 0.5, \mu_1(s_2) = 0.5, \mu_2(s_3) = 1$, $\mu_3(s_5) = 1$ and $\mu_4(s_{s_7}) = 1$. We have $|\pi| = 4$, and $\pi[2] = s_3$. It is easy to check that $\mathcal{P}$ is deterministic.*

An adversary $\mathcal{D}$ of PA $\mathcal{P}$ maps a finite path π of $\mathcal{P}$ to a pair (a, μ) or to $\curlywedge$, such that if $\mathcal{D}(\pi) = (a, \mu)$ for some $a \in Act$ and $\mu \in Dist(S)$, then $last(\pi) \xrightarrow{a} \mu$, and if there is no $a \in Act$ and $\mu \in Dist(S)$ s.t. $last(\pi) \xrightarrow{a} \mu$ then $\mathcal{D}(\pi) = \curlywedge$, where $\curlywedge \notin Act$ denotes the terminal action. We restrict the class of adversaries to the class of *admissible history-independent adversaries* [60]. An adversary $\mathcal{D}$ is called history-independent iff $last(\pi) = last(\pi') \Rightarrow \mathcal{D}(\pi) = \mathcal{D}(\pi')$ for any finite paths π and π'. Since our interest is in probabilistic reachability of PAs, it is sufficient to

consider history-independent adversaries [9]. Such adversaries are also known as memoryless schedulers. Admissible adversaries avoid the problem of unrealistic upper and lower bounds for the probability values [60]. This problem of unrealistic upper and lower bounds can be understood from the following example:

Example 2.26 *[61] Consider two players modeled by PA $\mathcal{P}$ (Fig. 2.4 (left)) and PA $\mathcal{P}'$ (Fig. 2.4 (middle)), are playing a game. Player $\mathcal{P}$ tosses a fair coin, waits a bit, then announces publicly that he is going to reveal the result of tossing (heads or tails), and then reveals the result. Player $\mathcal{P}'$ waits a bit, makes a guess about the result of the coin-tossing by player $\mathcal{P}$, then announces to reveal the result, and finally reveals it. The parallel composition of $\mathcal{P}$ and $\mathcal{P}'$ is shown in Fig. 2.4 (right). The probability that $\mathcal{P}'$ makes a correct guess is $\frac{1}{2}$. However, $\mathcal{P}||\mathcal{P}'$ in Fig. 2.4 (right) does not suggest this probability. In order to obtain the probabilities with which action ω (correct guess) is reported, adversaries or schedulers are used to resolve the nondeterminism. There are four possible schedulers for $\mathcal{P}||\mathcal{P}'$, yielding the set $\{0, \frac{1}{2}, 1\}$ of values of probabilities to observe action ω, which is incorrect. Admissible adversaries can be used to overcome this problem by only considering a subset of adversaries for computing reachability probabilities.*

More formally an admissible adversary is defined as follows:

Definition 2.27 *[60] An adversary is admissible if for any two finite paths π_1 and π_2 we have*

$$trace(\pi_1) = trace(\pi_2) \wedge last(\pi_1) \sim last(\pi_2) \implies \mathcal{D}(\pi_1) \equiv_{\sim} \mathcal{D}(\pi_2)$$

Intuitively, the definition of a admissible scheduler enforces that in cases when the adversary has observed the same history (given by the traces of the paths) and is in bisimilar states (denoted by $\sim$), it must schedule "the same" transitions up to bisimilarity [139]. Here $\equiv_{\sim}$ is an equivalence on the set of possible transition.

For PA $\mathcal{P}$, let $Paths^{\mathcal{D}}_{fin}(\mathcal{P})$ be the set of all finite paths, and $Paths^{\mathcal{D}}_{inf}(\mathcal{P})$ the set of all infinite paths of $\mathcal{P}$ under $\mathcal{D}$ that start in state s_0. Let $Adv(\mathcal{P})$ be the set of all admissible history-independent adversaries of PA $\mathcal{P}$. Let $Paths^{\mathcal{D},S_f}_{fin}(\mathcal{P})$ be the set of all finite paths under $\mathcal{D}$ that start in state s_0 and end in some state $s \in S_f$. For $\mathcal{D} \in Adv(\mathcal{P})$ let the probability measure $Prob^{\mathcal{D}}$ be defined over $Paths^{\mathcal{D}}_{inf}(\mathcal{P})$ in the following way. Let function $\mathbb{A} : Paths^{\mathcal{D}}_{fin}(\mathcal{P}) \times Paths^{\mathcal{D}}_{fin}(\mathcal{P}) \to [0, 1]$ be defined for two finite paths π, $\pi' \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$:

$$\mathbb{A}(\pi, \pi') = \begin{cases} \mu(s') & \text{if } \pi' \text{ is of the form } \pi \xrightarrow{a,\mu} s' \text{ and } \mathcal{D}(\pi) = (a, \mu) \\ 0 & \text{otherwise.} \end{cases}$$

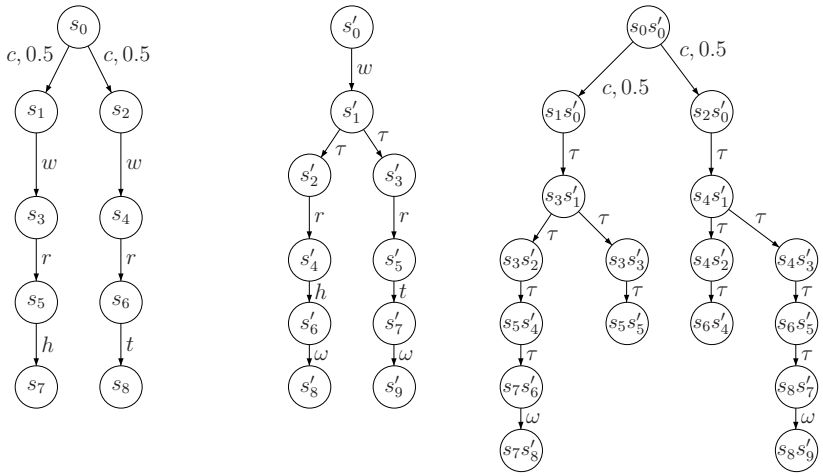

Figure 2.4: PAs $\mathcal{P}$ (left), $\mathcal{P}'$ (middle) and $\mathcal{P}||\mathcal{P}'$ (right)

The probability $P^{\mathcal{D}}(\pi)$ for any finite path $\pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$ where $|\pi| = n$ is now defined as follows:

$$P^{\mathcal{D}}(\pi) = \begin{cases} 1 & \text{if } n = 0 \\ \mathbb{A}(\pi[0], \pi[1]) \cdot \ldots \cdot \mathbb{A}(\pi[n-1], \pi[n]) & \text{otherwise.} \end{cases}$$

The cylinder of a finite path π is defined as follows:

$$cyl^{\mathcal{D}}(\pi) = \{\pi' \in Paths^{\mathcal{D}}_{inf}(\mathcal{P}) \mid \pi \text{ is a prefix of } \pi'\},$$

and let $\mathcal{F}^{\mathcal{D}}$ is the smallest σ-field containing $\{cyl^{\mathcal{D}}(\pi) \mid \pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})\}$. This all provides the basis to define $Prob^{\mathcal{D}}$ on $\mathcal{F}^{\mathcal{D}}$ as the unique measure such that $Prob^{\mathcal{D}}(cyl^{\mathcal{D}}(\pi)) = P^{\mathcal{D}}(\pi)$ for all $\pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$.

2.2.4 Interactive Markov Chains

IMCs [74, 75] extend LTSs with stochastic aspects. IMCs thus support both reasoning about nondeterministic behaviors as in LTSs and stochastic phenomena as in CTMCs. This section presents the basic concepts of IMCs.

Definition 2.28 (IMC) *An* interactive Markov chain *(IMC) is a tuple* $\mathcal{I} = (S, s_0, Act, AP, \rightarrow, \Rightarrow, L)$ *where:*

- S, s_0, Act, AP *and* L *are defined as before,*
- $\rightarrow \subseteq S \times Act \times S$ *is a set of interactive transitions,*

- $\Rightarrow \subseteq S \times \mathbb{R}_{\geq 0} \times S$ *is a set of Markovian transitions.*

We abbreviate $(s, a, s') \in \ \rightarrow$ as $s \xrightarrow{a} s'$ and similarly, $(s, \lambda, s') \in \ \Rightarrow$ by $s \overset{\lambda}{\Rightarrow} s'$. Let $IT(s)$ and $MT(s)$ denote the set of interactive and Markovian transitions that leave s. A state is *Markovian* iff $MT(s) \neq \varnothing$ and $IT(s) = \varnothing$; it is *interactive* iff $MT(s) = \varnothing$ and $IT(s) \neq \varnothing$. Further, s is a *hybrid* state iff $MT(s) \neq \varnothing$ and $IT(s) \neq \varnothing$; finally s is a *deadlock* state iff $MT(s) = \varnothing$ and $IT(s) = \varnothing$. Let $MS \subseteq S$ and $IS \subseteq S$ denote the set of Markovian and interactive states in IMC $\mathcal{I}$. For any Markovian state $s \in MS$ let $R(s, s') = \sum\{\lambda | s \overset{\lambda}{\Rightarrow} s'\}$ be the rate to move from state s to state s'. The exit rate for state s is defined by: $E(s) = \sum_{s' \in S} R(s, s')$.

It is easy to see that an IMC where $MT(s) = \varnothing$ for any state s is an LTS. An IMC where $IT(s) = \varnothing$ for any state s is a CTMC. The semantics of IMCs can thus be given in terms of the semantics of CTMCs (for Markovian transitions) and LTSs (for interactive transitions). An IMC is said to be closed if it is not subject to any further synchronization. We assume that in closed IMCs all outgoing interactive transitions of state s are labeled with $\tau \in Act$ (internal action).

Definition 2.29 *(Maximal progress) In any closed IMC, interactive transitions take precedence over Markovian transitions.*

Intuitively, the maximal progress assumption states that in closed IMCs, τ labeled transitions are not subject to interaction and thus can happen immediately, whereas the probability of a Markovian transition to happen immediately is zero. Accordingly, we assume that each state s has either only outgoing τ transitions or outgoing Markovian transitions. In other words, a closed IMC only has interactive and Markovian states.

Example 2.30 *Consider the IMC $\mathcal{I}$ shown in Fig. 2.5 where $AP = \{p, q, r\}$, $Act = \{a, b, c\}$ and s_0 is the initial state. The set of interactive states is $IS = \{s_0, s_1, s_2\}$; MS contains all other states. Nondeterminism between action transitions appears in state s_0.*

2.3 Modal Specification Theories

Specification theories are useful for the design and analysis of component-based systems in a top-down manner, from abstract specifications to implementations. A good specification theory should therefore support the notions of *satisfaction*

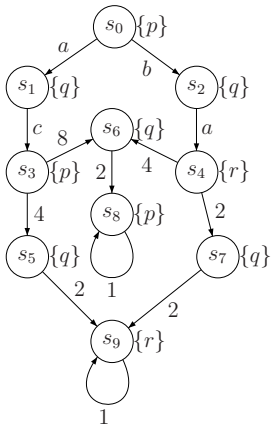

Figure 2.5: An example IMC $\mathcal{I}$

(to check whether an implementation satisfies a specification), *refinement* (to compare specifications w.r.t. their implementations) and *composition* (to combine specifications). Some examples of classical specification formalisms that we have studied in the previous sections are LTSs, PAs and IMCs. Note that all these models are compositional and support refinement through simulation relations. In LTSs and PAs, specification and implementation represent the same object. In this chapter we discuss the basic concepts of specification theories induced by modal transition systems (MTSs) and abstract probabilistic automata (APAs). MTSs are strictly more expressive than LTSs. For probabilistic systems that support nondeterminism, APAs provide a complete specification theory which is strictly more expressive than PAs.

2.3.1 Modal Transition Systems

Modal transition systems (MTSs) [102, 106] are labeled transition systems (LTSs) [9, 113] equipped with two types of transitions: *may* transitions that any implementation (LTS) may (or may not) have and *must* transitions that any implementation must have. An LTS is an MTS where all the transitions are must transitions. Next, we recall the basic concepts of modal transition systems with a finite state space. The definitions of satisfaction and refinement can be found in Section 6.1.

Definition 2.31 (MTS) *A modal transition system (MTS) is a tuple* $\mathcal{M} = (S, Act, s_0, S_f, V)$ *where:*

- *S, Act, s_0 and S_f are defined as before,*
- *$V : S \setminus S_f \times Act \times S \to \mathbb{B}_3$ is a three-valued transition function.*

Here $\mathbb{B}_3 = \{\bot, ?, \top\}$ denotes a complete lattice with the following ordering $\bot < ? < \top$ and meet $\sqcap$ and join $\sqcup$ operators. $V(s, a, s')$ identifies the a-labeled transition of the MTS in state s: $\top$, $?$ and $\bot$ indicate a *must*, a *may* and absence of such transition respectively. For simplicity we write $s \xrightarrow{a}_{\top} s'$ instead of $V(s, a, s') = \top$. Similarly, we write $s \xrightarrow{a}_{?} s'$ instead of $V(s, a, s') = ?$. Let $act(s)$ denote the set of enabled actions from state s, i.e., $act(s) = \{a \in Act \mid \exists s' : V(s, a, s') \neq \bot\}$. MTS $\mathcal{M}$ is *deterministic*, if for every state s and action a we have: $|\{s' \in S \mid V(s, a, s') \neq \bot\}| \leq 1$.

In this thesis we only consider deterministic MTSs, as they are sufficient for modeling the behavior of typical distributed algorithms [82]. An execution ρ of an MTS $\mathcal{M}$ is a (possibly infinite) sequence of the form $\rho = s_0 a_1 s_1 a_2 s_2 a_3 \ldots$, where $\forall n : s_n \xrightarrow{a_{n+1}}_{\top} s_{n+1}$ or $s_n \xrightarrow{a_{n+1}}_{?} s_{n+1}$. Let $Exec_{fin}(\mathcal{M})$ be the set of all finite executions, and $Exec_{inf}(\mathcal{M})$ the set of all infinite executions of $\mathcal{M}$ that start in state s_0. Let $Exec_{fin}^{S_f}(\mathcal{M})$ be the set of all finite executions of $\mathcal{M}$ that start in state s_0, and end in some state $s \in S_f$. Let $|\rho|$ be the length (number of actions) of a finite execution ρ. For infinite execution ρ and any $i \in \mathbb{N}$, let $\rho[i] = s_i$, the $(i+1)$-st state of ρ. For finite execution ρ of length n, $\rho[i]$ is only defined for $i \leq n$ and defined as for infinite executions. Let $last(\rho)$ denote the last state of ρ (if ρ is finite). Similarly, let $first(\rho)$ denote the first state of ρ.

2.3.2 Abstract Probabilistic Automata

Abstract probabilistic automata (APAs) [44, 45] have been proposed as a powerful specification and abstraction formalism for sets of PAs. In an APA, sets of distributions are abstracted by constraint functions. Action-labeled transitions of an APA are typed either "must" or "may". Hence, APAs can be seen as a combination of modal transition systems (MTSs) [106] and constraint Markov chains (CMCs) [30]. The theory of APAs is equipped with parallel and conjunction operators, and allows comparing two APAs using a refinement relation. A satisfaction relation is used to check whether a PA is an implementation of a given APA. Next, we recall the basic concepts of abstract probabilistic automata with a finite state space. The definitions of satisfaction and refinement can be found in Section 7.1.

Let φ be an arithmetic expression over variables whose values are in S. We

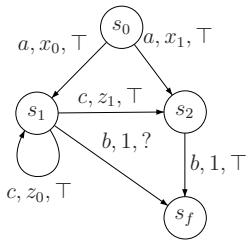

Figure 2.6: An example APA $\mathcal{N}$

call φ a *constraint function*. We require that for every variable in the arithmetic expression of φ, there exists a distribution μ that satisfies φ s.t. the value of the variable is nonzero. For example, we do not allow constraint function $\varphi_x = x_1 \geq 0.7 \wedge x_2 \leq 0.3 \wedge x_0 = 0 \wedge x_0 + x_1 + x_2 = 1$, since for every distribution μ that satisfies φ_x, the value of $x_0 = 0$. Let $C(S)$ be the set of all allowed constraint functions defined on S. Let $Sat(\varphi)$ denote the set of distributions that satisfy constraint function φ. Note that we do not restrict ourselves to linear constraint functions, as polynomial constraints are needed for defining layered and parallel composition (Chapter 7, Def. 7.11, Def. 7.14).

Definition 2.32 (APA) *An abstract probabilistic automaton (APA) is a tuple* $\mathcal{N} = (S, s_0, S_f, Act, V)$ *such that:*

- S, s_0, S_f *and* Act *are defined as before,*
- $V : S \setminus S_f \times Act \times C(S) \to \mathbb{B}_3$ *is a three-valued state-constraint function.*

Here $\mathbb{B}_3 = \{\bot, ?, \top\}$ denotes a complete lattice with the following ordering $\bot < ? < \top$ and meet $\sqcap$ and join $\sqcup$ operators as for MTSs. $V(s, a, \varphi)$ identifies the a-labeled transition of the APA in state s: $\top$, $?$ and $\bot$ indicate a *must*, a *may* and absence of transition respectively. For simplicity we write $s \xrightarrow{a}_{\top} \varphi$ instead of $V(s, a, \varphi) = \top$. Similarly, we write $s \xrightarrow{a}_{?} \varphi$ instead of $V(s, a, \varphi) = ?$. Let $act(s)$ denote the set of enabled actions from state s, i.e., $act(s) = \{a \in Act | \exists \varphi : V(s, a, \varphi) \neq \bot\}$.

Note that an APA where every transition is a must-transition and for each constraint function φ, the number of distributions in $Sat(\varphi)$ equals one, i.e., $|Sat(\varphi)| = 1$ is a PA [139, 141]. In simple words, every PA is an APA. Similarly an APA where every may and must transition jump to the next state with probability one is a modal transition system [106]. APA $\mathcal{N}$ is *deterministic*, if for every state s and action a we have: $|\{\varphi \in C(S) \mid V(s, a, \varphi) \neq \bot\}| \leq 1$.

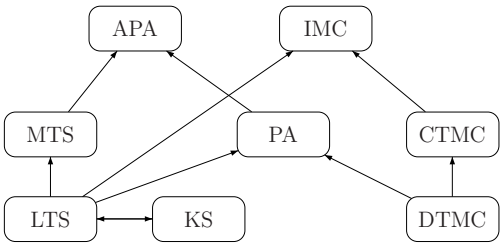

Figure 2.7: Relationship between nondeterministic and probabilistic models

Like for MTSs, we only consider deterministic APAs. Let $(a, \varphi)(s)$ denote the set of states in a deterministic APA $\mathcal{N}$ that can be reached from state s in one step by performing action a with constraint φ. Formally, $(a, \varphi)(s) = \{s' \in S \mid V(s, a, \varphi) \neq \bot \wedge \exists \mu \in Sat(\varphi) : \mu(s') > 0\}$. An abstract execution ρ of an APA $\mathcal{N}$ is a (possibly infinite) sequence of the form $\rho = s_0 a_1 \varphi_1 s_1 a_2 \varphi_2 s_2 a_3 \varphi_3 \ldots$, where $\forall n : s_n \xrightarrow{a_{n+1}} \top\varphi_{n+1}$ or $s_n \xrightarrow{a_{n+1}} ?\varphi_{n+1}$, and $s_{n+1} \in (a_{n+1}, \varphi_{n+1})(s_n)$. Let $Exec_{fin}(\mathcal{N})$ be the set of all finite abstract executions, and $Exec_{inf}(\mathcal{N})$ the set of all infinite abstract executions of $\mathcal{N}$ that start in state s_0. Let $Exec_{fin}^{S_f}(\mathcal{N})$ be the set of all finite abstract executions of $\mathcal{N}$ that start in state s_0, and end in some state $s \in S_f$. Let $Exec_{fin}^{s}(\mathcal{N})$ be the set of all finite abstract executions that start in state s, and $|\rho|$ the length (number of actions) of a finite abstract execution ρ. For infinite abstract execution ρ and any $i \in \mathbb{N}$, let $\rho[i] = s_i$, the $(i+1)$-st state of ρ. For finite abstract execution ρ of length n, $\rho[i]$ is only defined for $i \leq n$ and defined as in the case of infinite abstract executions. Let $last(\rho)$ denote the last state of ρ (if ρ is finite).

Example 2.33 *Consider the APA $\mathcal{N}$ in Fig. 2.6, where $S = \{s_0, s_1, s_2, s_f\}$, $Act = \{a, b, c\}$, s_0 is the initial state, and $S_f = \{s_f\}$. Here s_0 has one outgoing transition: a must a-transition (s_0, a, φ_x). Similarly, s_1 has two outgoing transitions: a must c-transition (s_1, c, φ_z), and a may b-transition $(s_1, b, 1)$. Note that $\mathcal{N}$ is deterministic. An example finite abstract execution ρ is $s_0 a \varphi_x s_1 c \varphi_z s_1 c \varphi_z s_2$ with $|\rho| = 3$, and $\rho[2] = s_1$.*

2.4 Summary

This chapter presented the basic concepts of a range of nondeterministic and probabilistic models. The relationship between these models can be understood

from Fig. 2.7. Here $A \to B$ denote that model B is an extension of model A.

Chapter 3

A Two Step Perspective for Kripke Structure Reduction

In this chapter we define Kripke minimization equivalence (KME) [142] and show that it allows for a more aggressive state space reduction than strong bisimulation for KSs, while preserving an interesting set of qualitative properties. In the weak setting we define weak Kripke minimization equivalence (WKME) such that state space reduction under WKME can potentially be larger than for divergence-sensitive stutter bisimulation. Whereas bisimulation compares states on the basis of their direct successors, KME considers a *two-step* perspective. The main principle is captured in Fig. 3.1 where boxes denote equivalence classes. Here states s_1 and s_2 are related, as for each pair of direct predecessors of the equivalence class $[s_1] = [s_2]$, i.e., in this case only s_0, the same set of equivalence classes, i.e., C_1, C_2 and C_3, can be reached in two steps via $[s_1]$. For WKME, we abstract from stutter steps and thus each predecessor of any equivalence class C should reach the same set of equivalence classes in two or more steps such that all extra steps are taken within C.

Contributions. The main contributions of this chapter are as follows:

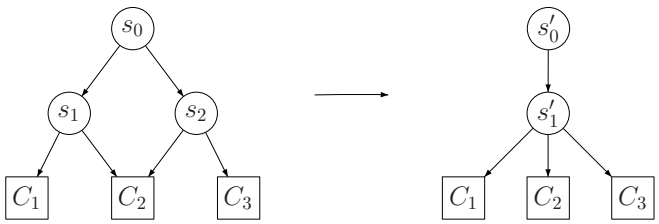

Figure 3.1: KS aggregation under Kripke minimization equivalence

- We provide a structural definition of KME on KSs, define the quotient under KME and show that KME is strictly coarser than strong bisimulation.
- We show that linear-time (LT) properties defined over infinite words are preserved under KME quotienting.
- Next, we show that KME is compositional w.r.t. synchronous parallel compositon (SCCS-like parallel composition [112]).
- In the weak setting, we provide a structural definition of WKME on KSs, define the quotient under WKME and show that WKME is strictly coarser than divergence-sensitive stutter bisimulation.
- Finally, we prove that stutter-insensitive LT properties defined over infinite words are preserved under WKME quotienting.

Organisation of this chapter. Section 3.1 defines Kripke minimization equivalence and discusses the preservation of LT properties under KME quotienting. In Section 3.2, we prove that KME is compositional w.r.t. synchronous parallel composition. Sections 3.3 defines weak Kripke minimization equivalence and discusses the preservation of stutter-insensitive LT properties under WKME quotienting. Section 3.4 discusses related work. Finally, Section 3.5 concludes the chapter.

3.1 Kripke Minimization Equivalence

In this section, we present a technique for the state space minimization of a KS. We first define *Kripke minimization equivalence* (KME) followed by the definition of quotient KS under KME. Next to that, the relationship between KME and strong bisimulation is explored. All the definitions in this section are relative to a KS $\mathcal{K} = (S, \rightarrow, AP, L, s_0)$.

Definition 3.1 (Predecessor based reachability) *For $s \in S$ and $C, D \subseteq S$, the function $Pbr : S \times 2^S \times 2^S \rightarrow \{0, 1\}$ is defined as:*

$$Pbr(s, C, D) = \begin{cases} 1 & \text{if } \exists s' \in Post(s, C) \text{ s.t. } Post(s', D) \neq \varnothing \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.2 (KME) *Equivalence $\mathcal{R}$ on S is a* Kripke minimization equivalence *(KME) on KS $\mathcal{K}$ if we have:*

1. $\forall (s_1, s_2) \in \mathcal{R}$ *it holds:* $L(s_1) = L(s_2)$ *and*
2. $\forall C, D \in S/_{\mathcal{R}}$ *and* $\forall s', s'' \in Pred(C)$ *it holds:* $Pbr(s', C, D) = Pbr(s'', C, D)$

States s_1, s_2 *are* KM *related, denoted by* $s_1 \star s_2$, *if* $(s_1, s_2) \in \mathcal{R}$ *for some KME* $\mathcal{R}$.

Remark 3.3 *Note that* $\star$ *is not a KME, this is contrary to what one usually expects from a coinductive/bisimulation-style definition.*

Example 3.4 *Consider the KS* $\mathcal{K}$ *in Fig. 3.2 (left). Let* $C = \{s_3, s_4, s_5\}$ *and* $D = \{s_7\}$. *Then* $Pbr(s_1, C, D) = 1$, *since it is possible to move from* s_1 *to* s_7 *in two steps via* s_3. *Similarly* $Pbr(s_2, C, D) = 1$. *For KS* $\mathcal{K}$, *the equivalence relation induced by the partitioning* $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6\}, \{s_7\}\}$ *is a KME.*

3.1.1 Quotient Kripke Structure

Definition 3.5 (Quotient Kripke structure) *For KME relation* $\mathcal{R}$ *on KS* $\mathcal{K}$, *the quotient KS is defined by* $\mathcal{K}/_{\mathcal{R}} = (S/_{\mathcal{R}}, \rightarrow', AP, L', s_0')$ *where:*

- $S/_{\mathcal{R}}$ *is the set of all equivalence classes under* $\mathcal{R}$,
- $\rightarrow' \subseteq S/_{\mathcal{R}} \times S/_{\mathcal{R}}$ *is defined by:* $C \rightarrow' D$ *iff* $Pbr(s', C, D) = 1$ *where* $s' \in Pred(C)$ *and* $C, D \in S/_{\mathcal{R}}$,
- $L'(C) = L(s)$, *where* $s \in C$ *and*
- $s_0' = C$ *where* $s_0 \in C = [s_0]_{\mathcal{R}}$.

Example 3.6 *The quotient KS for the Fig. 3.2 (left) under the KME relation* $\mathcal{R}$ *induced by the partitioning* $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6\}, \{s_7\}\}$ *is shown in Fig. 3.2 (right).*

Definition 3.7 *KS* $\mathcal{K}$ *and its quotient* $\mathcal{K}/_{\mathcal{R}}$ *under KME relation* $\mathcal{R}$ *are* $\star$*-related, denoted by* $\mathcal{K} \star \mathcal{K}/_{\mathcal{R}}$, *if and only if there exists a KME relation* $\mathcal{R}^*$ *defined on the disjoint union* $S \uplus S/_{\mathcal{R}}$ *such that*

$$\forall C \in S/_{\mathcal{R}} \colon \forall s \in C \implies (s, C) \in \mathcal{R}^*.$$

Theorem 3.8 *For any KS* $\mathcal{K}$ *and KME* $\mathcal{R}$ *on* $S : \mathcal{K} \star \mathcal{K}/_{\mathcal{R}}$.

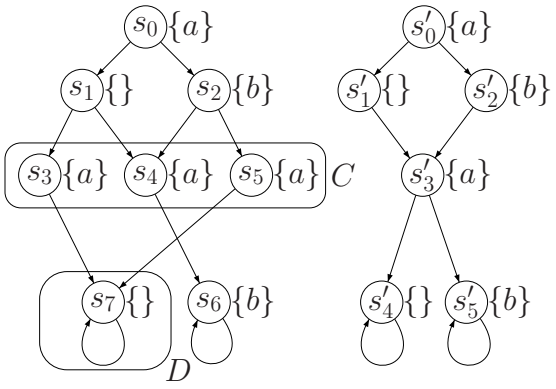

Figure 3.2: KS $\mathcal{K}$ (left) and its quotient $\mathcal{K}/_{\mathcal{R}}$ under a KME (right)

Remark 3.9 *Note that union of KMEs is not necessarily a KME. In other words, it is possible that $\mathcal{R}_1, \mathcal{R}_2$ are two KMEs on S s.t. $\mathcal{R}_1 \cup \mathcal{R}_2$ is not a KME. Intuitively it means that the original KS $\mathcal{K}$ can be reduced in different ways. This can be observed from the example given below.*

Example 3.10 *Consider the KS $\mathcal{K}$ shown in Fig. 3.3. In this case, $\mathcal{K}$ can be minimized in two different ways as shown in Fig. 3.4. Here KS $\mathcal{K}'$ shown in Fig. 3.4 (left) is the quotient system for the KME relation $\mathcal{R}$ induced by $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6, s_7, s_8\}, \{s_9\}, \{s_{14}\}, \{s_{10}, s_{12}\}, \{s_{11}, s_{13}\}\}$. KS $\mathcal{K}''$ shown in Fig. 3.4 (right) is the quotient system for the KME $\mathcal{R}'$ induced by $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}, \{s_4, s_5, s_6, s_7\}, \{s_8\}, \{s_9\}, \{s_{14}\}, \{s_{10}, s_{12}\}, \{s_{11}, s_{13}\}\}$. It is easy to check that $\mathcal{R} \cup \mathcal{R}'$ is not a KME.*

Repeated minimization. Next, we show that KME can be used for repeated minimization of a KS. Intuitively, this means that if a quotient system $\mathcal{K}'$ has been obtained from a KS $\mathcal{K}$ under KME $\mathcal{R}$ on S, then it might still be possible to further reduce $\mathcal{K}'$ to $\mathcal{K}''$ under some KME $\mathcal{R}'$ on S'. Consider the KS shown in Fig. 3.5 (left). KS in Fig. 3.5 (middle) is the quotient system for the KME induced by the partition $\{\{s_0\}, \{s_1, s_2\}, \{s_3, s_4\}, \{s_5\}, \{s_6\}, \{s_7\}, \{s_8\}\}$. KS in Fig. 3.5 (right) is the quotient of the KS Fig. 3.5 (middle) for the KME induced by the parttion $\{\{s'_0\}, \{s'_1\}, \{s'_2, s'_3\}, \{s'_4\}, \{s'_5\}, \{s'_6\}\}$. It is easy to check that s_3, s_4, s_5 in the original system cannot be merged in one shot, since s_1 can reach states labeled with atomic propositions a and b in two steps via s_3 and s_4 respectively, but s_2 cannot reach such states. This is no longer a problem once s_1 and s_2 are

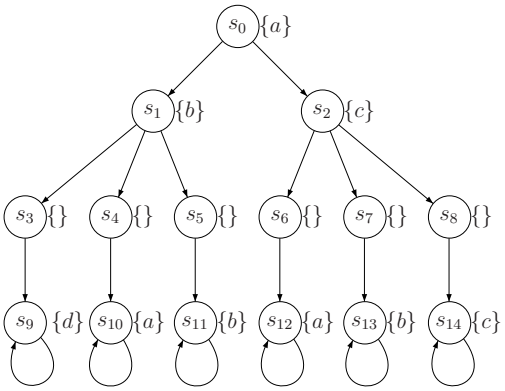

Figure 3.3: An example KS $\mathcal{K}$

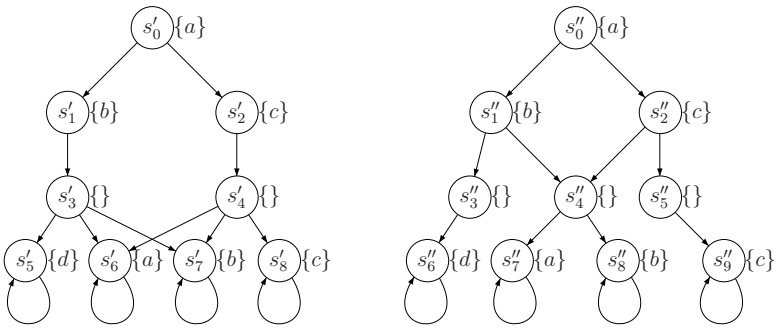

Figure 3.4: Union of KMEs is not necessarily a KME

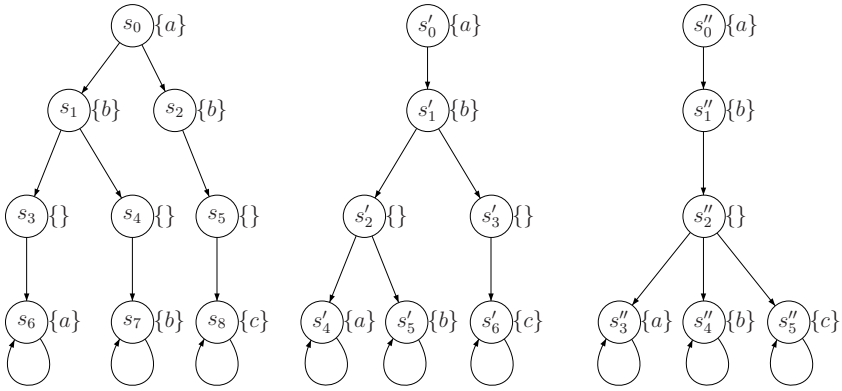

Figure 3.5: Repeated minimization

merged as shown in Fig. 3.5 (middle) as s'_2, s'_3 now have a single predecessor, i.e., s'_1.

3.1.2 KME vs. Bisimulation

Definition 3.11 (Strong bisimulation [9]) *Binary relation $\mathcal{R}$ on S is a* strong bisimulation *on $\mathcal{K}$ if for any $(s_1, s_2) \in \mathcal{R}$ we have:*

- $L(s_1) = L(s_2)$,
- *if $s'_1 \in Post(s_1)$ then there exists $s'_2 \in Post(s_2)$ with $(s'_1, s'_2) \in \mathcal{R}$, and*
- *if $s'_2 \in Post(s_2)$ then there exists $s'_1 \in Post(s_1)$ with $(s'_1, s'_2) \in \mathcal{R}$.*

States s_1, s_2 are bisimilar*, denoted $s_1 \sim s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some strong bisimulation $\mathcal{R}$.*

These conditions require that any two bisimilar states, say s_1, s_2 are equally labeled and that every outgoing transition of s_1 must be matched by an outgoing transition of s_2 and vice versa. Note that the relation $\sim$ is an equivalence relation and is the coarsest strong bisimulation.

Theorem 3.12 *KME is strictly coarser than $\sim$.*

This theorem says that state space reduction under KME can potentially be larger than for strong bisimilarity.

For strong simulation equivalence, the condition to exhibit identical stepwise behavior is slightly relaxed. Whenever s' simulates s, state s' can mimic all stepwise behavior of s; the reverse is not guaranteed, so state s' may perform transitions that cannot be matched by state s. Two Kripke structures $\mathcal{K}$ and $\mathcal{K}'$ are simulation-equivalent if their initial states mutually simulate each other.

Remark 3.13 *Consider the two KSs in Fig. 3.2, here $\mathcal{K}$ and $\mathcal{K}/_{\mathcal{R}}$ are not strong simulation equivalent. This shows that there are systems that can be reduced using KME, but cannot be reduced under simulation equivalence. It would be interesting to investigate if the proof of Thm. 3.12 can be extended showing that the quotient obtained under simulation equivalence can be obtained by repeated application of KME. In simple words, this would mean that if $\mathcal{K}'$ can be obtained from $\mathcal{K}$ under simulation equivalence, then $\mathcal{K}'$ can also be obtained by repeated application of KME.*

3.1.3 Property Preservation

Linear-time properties. We investigate linear-time properties for KSs that are preserved under KME quotienting. We study a more general class of linear-time properties that are defined over traces, i.e., $(2^{AP})^{\omega}$. These include, e.g., ω-regular properties. Note that the preservation of ω-regular properties implies the preservation of LTL formulas. These preservation results can be exploited for model checking by reducing the KS models under consideration prior to carrying out the verification.

Definition 3.14 *A* linear-time *property (LT property) over the set of atomic propositions AP is a subset of $(2^{AP})^{\omega}$.*

Example 3.15 *An LT property can be used to specify the desired behavior of the system under consideration such as:*

- *Every time the process tries to send a message, it eventually succeeds in sending it.*
- *Whenever the system is down, an alarm should ring until it is up again.*

Definition 3.16 *Let P be an LT property over AP and $\mathcal{K} = (S, \rightarrow, AP, L, s_0)$ a Kripke structure. Then $\mathcal{K}$ satisfies P, denoted $\mathcal{K} \models P$, iff $Traces^{\mathcal{K}}(s_0) \subseteq P$.*

Theorem 3.17 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ be a KME on $\mathcal{K}$. Then for any LT property $P : \mathcal{K} \models P \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models P$.*

Intuitively, this theorem says that if an LT property holds for the original Kripke structure, it also holds for the quotient and vice versa. In principle this result allows performing model checking on the quotient Kripke structure provided that we can obtain this in an algorithmic manner.

Corollary 3.18 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ be a KME on $\mathcal{K}$. Then for any LTL formula $\varphi : \mathcal{K} \models \varphi \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models \varphi$.*

3.2 Synchronous Parallel Composition

In this section we show that KME is compositional w.r.t. synchronous parallel composition (SCCS-like parallel composition [112]) of KSs. This result is useful for analyzing synchronous distributed algorithms and synchronous hardware circuits where processes progress in a lock-step fashion. For example say we want to compose a large KS $\mathcal{K}_1$ with another KS $\mathcal{K}_2$ and these KSs have n and m states respectively. Then the resulting KS $\mathcal{K}_1 \otimes \mathcal{K}_2$ can have $n \cdot m$ states (in the worst case) so it is worthwhile to compute this composition using a smaller KS $\mathcal{K}'$ $\star$-related to $\mathcal{K}_1$. Synchronous parallel composition is also at the heart of Lustre [72], a declarative programming language for reactive systems, and is used in many other hardware-oriented languages.

Definition 3.19 *[112] Let $\mathcal{K}_1 = (S_1, \rightarrow_1, AP_1, L_1, s_{01})$ and $\mathcal{K}_2 = (S_2, \rightarrow_2, AP_2, L_2, s_{02})$ be two Kripke structures. Then we say $s \rightarrow_i s'$ if $(s, s') \in \rightarrow_i$ for $i = 1, 2$. The synchronous parallel composition of two Kripke structures is $\mathcal{K}_1 \otimes \mathcal{K}_2 = (S_1 \times S_2, \rightarrow, AP_1 \cup AP_2, L, (s_{01}, s_{02}))$, where (s_{01}, s_{02}) is the initial state, $L((s_1, s_2)) = L(s_1) \cup L(s_2)$, and $\rightarrow$ is given as follows:*

$$\frac{s_1 \rightarrow_1 s_1' \wedge s_2 \rightarrow_2 s_2'}{(s_1, s_2) \rightarrow (s_1', s_2')}.$$

Note that $\otimes$ is commutative and associative.

Theorem 3.20 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ be a KME on $\mathcal{K}$. Then for any Kripke structure $\mathcal{K}_1$:*

$$(\mathcal{K} \otimes \mathcal{K}_1) \star (\mathcal{K}/_R \otimes \mathcal{K}_1).$$

3.3 Weak Kripke Minimization Equivalence

In this section we define weak Kripke minimization equivalence (WKME). WKME is a variant of KME that abstracts from stutter steps, also referred to as internal

or nonobservable steps. To compare KSs that model a given system at different abstraction levels, it is often too demanding to require a statewise equivalence. Instead, a state in a KS at a high level of abstraction can be modeled by a sequence of states in the more concrete KS. Secondly, by abstracting from internal steps, quotient KSs are obtained that may be significantly smaller than the quotient under corresponding strong equivalence relation. Interestingly, though, still a rather rich set of properties is preserved under such abstractions. More specifically, weak equivalences are suitable for verifying properties, for which the exact number of transitions a system takes to accomplish some task is irrelevant.

Definition 3.21 (Weak predecessor based reachability) *For $s \in S$ and $C, D \subseteq S$, the function $WPbr : S \times 2^S \times 2^S \to \{0,1\}$ is defined as:*

$$WPbr(s,C,D) = \begin{cases} 1 & \text{if } \exists s' \in Post(s,C), s'' \in D \text{ s.t.} s' \xrightarrow{*} s'' \\ 0 & \text{otherwise.} \end{cases}$$

where $s' \xrightarrow{*} s''$ denotes that there exists a path $\pi = s' \to s_1 \to s_2 \ldots s_n \to s''$, where $n \geq 0$ and $s_i \in C, i = 1, \ldots, n$.

Remark 3.22 *Note that if $n = 0$ then $s' \xrightarrow{*} s''$ denotes $s' \to s''$, i.e., one step reachability.*

Definition 3.23 (WKME) *Equivalence $\mathcal{R}$ on S is a* weak Kripke minimization equivalence *(WKME) on $\mathcal{K}$ if we have:*

1. *$\forall (s_1, s_2) \in \mathcal{R}$ it holds: $L(s_1) = L(s_2)$ and*
2. *$\forall C, D \in S/_{\mathcal{R}}$ s.t. $C \neq D$ and $\forall s', s'' \in Pred(C)$ s.t. $s', s'' \notin C$ it holds: $WPbr(s', C, D) = WPbr(s'', C, D)$.*

States s_1, s_2 are WKM *related, denoted by $s_1 \odot s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some WKME $\mathcal{R}$.*

Example 3.24 *Consider the KS $\mathcal{K}$ in Fig. 3.6. Let $C = \{s_3, s_4, s_5\}$ and $D = \{s_6\}$. Then $WPbr(s_1, C, D) = 1$, since it is possible to move from s_1 to s_6 in three steps via s_3, s_4 (where $s_3 \to s_4$ is a stutter step). Similarly $WPbr(s_2, C, D) = 1$. For KS $\mathcal{K}$, the equivalence relation induced by the partitioning $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6\}, \{s_7\}\}$ is a WKME relation.*

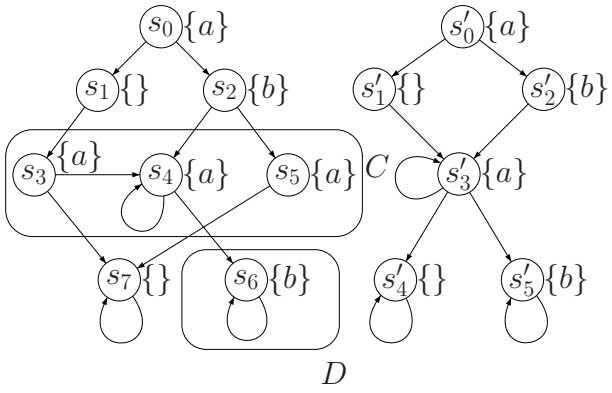

Figure 3.6: KS $\mathcal{K}$ (left) and its quotient $\mathcal{K}/_{\mathcal{R}}$ under a WKME (right)

3.3.1 Quotient Kripke Structure

Definition 3.25 (Quotient Kripke structure) *For WKME relation $\mathcal{R}$ on KS $\mathcal{K}$, the quotient KS is defined by $\mathcal{K}/_{\mathcal{R}} = (S/_{\mathcal{R}}, \rightarrow', AP, L', s'_0)$ where:*

- *$S/_{\mathcal{R}}$ is the set of all equivalence classes under $\mathcal{R}$,*
- *$\rightarrow' \subseteq S/_{\mathcal{R}} \times S/_{\mathcal{R}}$ is defined by: $C \rightarrow' D$, s.t. $C \neq D$ iff $WPbr(s', C, D) = 1$ where $s' \in Pred(C)$, and $C \rightarrow' C$ iff there exists $s \in C$ s.t. $s \xrightarrow{*} s$*
- *$L'(C) = L(s)$, where $s \in C$ and*
- *$s'_0 = C$ where $s_0 \in C = [s_0]_{\mathcal{R}}$.*

Example 3.26 *The quotient KS for the Fig. 3.6 (left) under the WKME relation $\mathcal{R}$ induced by the partitioning $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6\}, \{s_7\}\}$ is shown in Fig. 3.6 (right).*

Definition 3.27 *Any Kripke structure $\mathcal{K}$ and its quotient $\mathcal{K}/_{\mathcal{R}}$ under WKME relation $\mathcal{R}$ are $\odot$-related, denoted by $\mathcal{K} \odot \mathcal{K}/_{\mathcal{R}}$, if and only if there exists a WKME relation $\mathcal{R}^*$ defined on the disjoint union $S \uplus S/_{\mathcal{R}}$ such that*

$$\forall C \in S/_{\mathcal{R}} \colon \forall s \in C \implies (s, C) \in \mathcal{R}^*.$$

Theorem 3.28 *For any KS $\mathcal{K}$ and WKME $\mathcal{R}$ on $S : \mathcal{K} \odot \mathcal{K}/_{\mathcal{R}}$.*

Remark 3.29 *Note that WKMEs can be used for repeated minimization of a KS $\mathcal{K}$ and union of WKMEs is not necessarily a WKME.*

Theorem 3.30 *WKME is strictly coarser than KME.*

3.3.2 WKME vs. Divergence-Sensitive Stutter Bisimulation

Definition 3.31 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ an equivalence relation on S.*

- *$s \in S$ is $\mathcal{R}$-divergence-sensitive if there exists an infinite path fragment $\pi = s \to s_1 \to s_2 ... \in Paths(s)$ s.t. $(s, s_j \in \mathcal{R})$ for all $j > 0$.*
- *$\mathcal{R}$ is divergence-sensitive if for any $(s_1, s_2) \in \mathcal{R}$: if s_1 is $\mathcal{R}$-divergence-sensitive, then s_2 is $\mathcal{R}$-divergence-sensitive.*

Definition 3.32 *Divergence-sensitive relation $\mathcal{R}$ on S is a* stutter bisimulation *on $\mathcal{K}$ if for any $(s_1, s_2) \in \mathcal{R}$ we have:*

- *$L(s_1) = L(s_2)$,*
- *If $s_1' \in Post(s_1)$ with $(s_1', s_2) \notin \mathcal{R}$, then there exists a finite path fragment $s_2 \to u_1 \to \ldots u_n \to s_2'$ with $n \geq 0$ and $(s_1, u_i) \in \mathcal{R}$, $i = 1, \ldots, n$ and $(s_1', s_2') \in \mathcal{R}$,*
- *If $s_2' \in Post(s_2)$ with $(s_1, s_2') \notin \mathcal{R}$, then there exists a finite path fragment $s_1 \to v_1 \to \ldots v_n \to s_1'$ with $n \geq 0$ and $(v_i, s_2) \in \mathcal{R}$, $i = 1, \ldots, n$ and $(s_1', s_2') \in \mathcal{R}$.*

States s_1 and s_2 are divergence-sensitive stutter bisimilar*, denoted by $s_1 \cong^{div} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some divergence-sensitive stutter bisimulation $\mathcal{R}$.*

Next, we investigate the relationship between WKME and divergence-sensitive stutter bisimulation relation.

Theorem 3.33 *WKME is strictly coarser than $\cong^{div}$.*

This theorem asserts that WKME can achieve larger state space reduction as compared to divergence-sensitive stutter bisimulation.

For divergence-sensitive stutter simulation equivalence [116] the conditions provided in Def. 3.32 are slightly relaxed. Whenever s' stutter simulates s, state s' can stutter mimic all stepwise behavior of s, and if there exists a path π emanating from state s such that all the states on π are related to state s', then s' has to have some successor s_n' such that some state s_n on π is related to s_n', the reverse is not guaranteed, so state s' may perform transitions that cannot be stutter mimicked by state s. Two Kripke structures $\mathcal{K}$ and $\mathcal{K}'$ are divergence-sensitive stutter simulation-equivalent if their initial states mutually stutter simulate each other according to the conditions given above.

Remark 3.34 *Consider the two KSs in Fig. 3.6, here $\mathcal{K}$ and $\mathcal{K}/_{\mathcal{R}}$ are not divergence-sensitive stutter simulation equivalent. It would be interesting to investigate if the proof of Thm. 3.33 can be extended showing that the quotient obtained under divergence-sensitive stutter simulation equivalence can be obtained by repeated application of WKME.*

3.3.3 Property Preservation

Stutter-insensitive LT properties. We investigate stutter-insensitive LT properties defined over infinite words for KSs that are preserved under WKME quotienting. These include, e.g., stutter-insensitive ω-regular properties. Note that the preservation of stutter-insensitive ω-regular properties [129] implies the preservation of $LTL\backslash\bigcirc$ formulas.

Definition 3.35 *LT property P over AP is stutter-insensitive if for any $\rho \in P$, $\forall \rho_1 \in (2^{AP})^\omega$ s.t. $\rho_1 \triangleq \rho \Rightarrow \rho_1 \in P$.*

Example 3.36 *Consider the stutter-insensitive LT property [41]:*

$$\mathbf{P}_n := \{w \in (2^{\{p\}})^\omega : \textit{the number of occurrences of the sub-word } \{p\}\varnothing \textit{ in } w \textit{ is divisible by } n\},$$

for natural $n \geq 2$. This property cannot be expressed using $LTL\backslash\bigcirc$.

The satisfaction relation for stutter-insensitive LT property P, i.e., $\mathcal{K} \models P$, is as in Def. 3.16.

Theorem 3.37 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ be a WKME on $\mathcal{K}$. Then for any stutter-insensitive LT property $P : \mathcal{K} \models P \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models P$.*

Corollary 3.38 *Let $\mathcal{K}$ be a KS and $\mathcal{R}$ be a WKME on $\mathcal{K}$. Then for any $LTL\backslash\bigcirc$ formula $\varphi : \mathcal{K} \models \varphi \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models \varphi$.*

3.4 Related Work

For KSs, one usually distinguishes between linear-time and branching-time equivalence relations [64]. The standard example of a linear-time equivalence is trace equivalence [76, 135, 148]. Informally, two states are trace equivalent if the possible sequences of words starting from these states are the same. Several extensions

of trace equivalence have been proposed, e.g., failure trace semantics and readiness trace semantics [6, 26, 77, 118, 119, 122, 130, 132, 155]. In the weak setting, stutter trace equivalence has been proposed where a pair of sequences are considered to be equivalent if they differ in at most the number of times a set of propositions may adjacently repeat [101]. Checking trace equivalence is PSPACE-complete. In branching-time semantics, various relations on KSs have been defined such as strong and stutter variants of bisimulation and simulation pre-orders [27, 65, 68, 110, 111, 126]. Strong bisimulation and divergence-sensitive stutter bisimulation coincide with Computation Tree Logic (CTL^*) and $CTL^*\backslash_{\bigcirc}$, respectively [27, 120]. Strong simulation agrees with a "preorder" on the universal (or existential) fragment of CTL [36]. Several papers report data showing that bisimulation minimization can substantially reduce the state-space of models to be verified [5, 55]. The use of simulation relations for abstraction has been studied in, e.g. [36, 40, 108].

3.5 Conclusions

In this chapter, we have presented two equivalence relations, Kripke minimization equivalence (KME) and weak Kripke minimization equivalence (WKME) on KSs. We defined the quotient system under these relations and proved that these relations are coarser than strong bisimulation and divergence-sensitive stutter bisimulation, respectively. Preservation results for LT properties and stutter-insensitive LT properties have been established under KME and WKME quotienting. We have also shown that KME is compositional w.r.t. synchronous parallel composition.

3. A TWO STEP PERSPECTIVE FOR KRIPKE STRUCTURE REDUCTION

Chapter 4

Weighted Probabilistic Equivalence

This chapter defines weighted probabilistic equivalence (WPE) [143] and shows that it allows for a more aggressive state space aggregation than probabilistic bisimulation also known as lumping for DTMCs, while still preserving an interesting set of quantitative properties.

Unlike bisimulation that compares states on the basis of their direct successors —the cumulative probability to directly move to any equivalence class must be equal— WPE considers a *two-step* perspective. Two states s, s' are related if for each pair of their direct predecessors the weighted probability to directly move to any equivalence class via the equivalence class $[s] = [s']$ coincides. The main principle is captured in Fig. 4.1 where boxes denote equivalence classes, and $p_{C_1} = p'_1 \cdot p_{1,1}$, $p_{C_2} = p'_1 \cdot p_{1,2} + p'_2 \cdot p_{2,1}$, $p_{C_3} = p'_2 \cdot p_{2,2}$ with $p'_1 = \frac{p_1}{p_1+p_2}$ and $p'_2 = \frac{p_2}{p_1+p_2}$. Here states s_1 and s_2 are related, as for each pair of direct predecessors of the equivalence class $[s_1] = [s_2]$, i.e., in this case only s_0, the weighted probability to move to all the states in the equivalence class C_i (for i=1, 2, 3) via all the states in $[s_1]$ is equal. This allows combining states s_1 and s_2 into state s'_1, cf. the right DTMC in Fig. 4.1.

We provide a structural definition of WPE on DTMCs. We define the quotient under WPE, show that any DTMC is related to its quotient under WPE, and prove that WPE is (strictly) coarser than bisimulation. The main contributions of this chapter are as follows:

- We show that ω-regular properties are preserved under WPE quotienting.
- Next, we show that WPE is compositional w.r.t. synchronous parallel com-

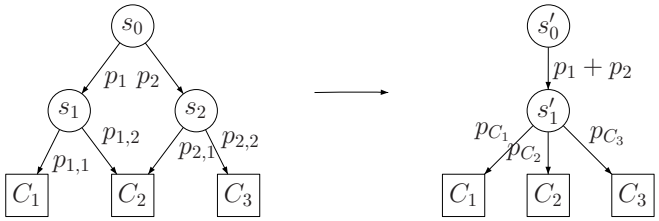

Figure 4.1: DTMC aggregation under weighted probabilistic equivalence

position of DTMCs.

- Finally, we extend these results to Markov reward models, i.e., DTMCs with rewards.

We first show that the probability of satisfying a deterministic Rabin automaton (DRA) [150, pp. 3-21], [9, pp. 801-805] specification for any DTMC coincides with the probability for its quotient. Since the class of languages accepted by DRAs agrees with the class of ω-regular properties it implies that WPE preserves ω-regular properties. We note that this also implies the preservation of Linear Temporal Logic (LTL) [9, pp. 229-270] formulas and transient-state probabilities. It is important to point out that there are certain interesting ω-regular properties, e.g., every even position should always be occupied by a, that cannot be expressed using PCTL, PCTL* (an extension of PCTL) or LTL. Model checking a DTMC against a DRA specification can be done by solving a system of linear equations obtained on the product of the DTMC and the DRA [9, pp. 803-805]. We also show that WPE is compositional w.r.t. synchronous parallel composition [151, 152]. This is helpful as instead of analyzing a large DTMC, which may be very costly, we can analyze the smaller, WPE related DTMC. Finally we extend these preservation results to Markov reward models.

Organisation of this chapter. Section 4.1 defines weighted probabilistic equivalence, treats some basic properties and discusses the preservation of ω-regular properties. Section 4.2 shows that WPE is compositional w.r.t. synchronous parallel composition. Section 4.3 presents the extension of WPE to Markov reward models. Section 4.4 discusses related work. Finally, Section 4.5 concludes the chapter.

4.1 Weighted Probabilistic Equivalence

This section presents the basic concepts related to *weighted probability* followed by the formal definition of weighted probabilistic equivalence. We also define the quotient DTMC under WPE and explore its relationship with bisimulation. All the definitions in this section are relative to a DTMC $\mathcal{D} = (S, P, AP, L, s_0)$.

Definition 4.1 *For $s, s' \in S$ and $C \subseteq S$, the function $P : S \times S \times 2^S \to \mathbb{R}_{\geq 0}$ is defined by:*

$$P(s, s', C) = \begin{cases} \frac{P(s,s')}{P(s,C)} & \text{if } s' \in C \text{ and } P(s, C) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, $P(s, s', C)$ is the probability to move from state s to s' under the condition that s moves to some state in C.

Remark 4.2 *Note that P is already used and in Def. 4.1 we overload P.*

Definition 4.3 (Weighted probability) *For $s \in S$, and $C, D \subseteq S$, the function $wp : S \times 2^S \times 2^S \to \mathbb{R}_{\geq 0}$ is defined by:*

$$wp(s, C, D) = \sum_{s' \in C} P(s, s', C) \cdot P(s', D).$$

Intuitively, $wp(s, C, D)$ is the (weighted) probability to move from s to some states in D in two steps via states of C.

Definition 4.4 (WPE) *Equivalence $\mathcal{R}$ on S is a* weighted probabilistic equivalence *(WPE) if we have:*

1. *$\forall (s_1, s_2) \in \mathcal{R}$ it holds: $L(s_1) = L(s_2)$, and*
2. *$\forall C, D \in S/_{\mathcal{R}}$ and $\forall s', s'' \in Pred(C)$ it holds: $wp(s', C, D) = wp(s'', C, D)$,*

States s_1, s_2 are WP *related, denoted by $s_1 \stackrel{\circ}{=} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some WPE $\mathcal{R}$.*

These conditions require that any two related states are equally labeled and that for any two equivalence classes $C, D \in S/_{\mathcal{R}}$, where $S/_{\mathcal{R}}$ denotes the set consisting of all $\mathcal{R}$-equivalence classes, the weighted probability of going from any two predecessors of C to D via any state in C must be equal. Note that, by definition, any WPE is an equivalence relation.

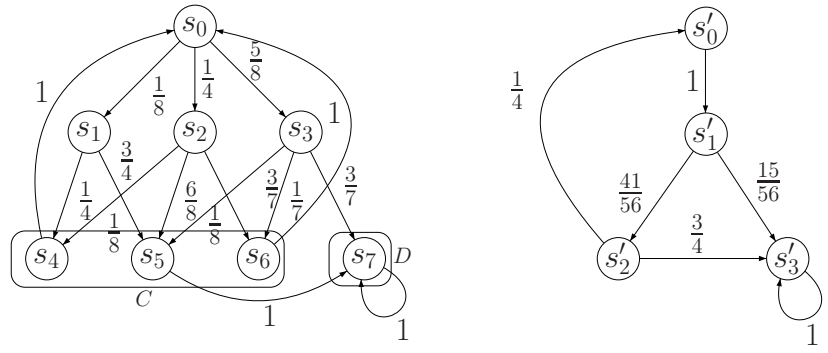

Figure 4.2: DTMC $\mathcal{D}$ (left) and its quotient under a WPE $\mathcal{D}/_{\mathcal{R}}$ (right)

Example 4.5 *Consider the DTMC $\mathcal{D}$ in Fig. 4.2 (left). Let $C = \{s_4, s_5, s_6\}$ and $D = \{s_7\}$. Let the labeling function for $\mathcal{D}$ be the same as in Example 2.12. Then* $wp(s_1, C, D)$

$$= \underbrace{P(s_1, s_4, C)}_{=\frac{1}{4}} \cdot \underbrace{P(s_4, D)}_{=0} + \underbrace{P(s_1, s_5, C)}_{\frac{3}{4}} \cdot \underbrace{P(s_5, D)}_{=1} = \frac{3}{4}.$$

Similarly, $wp(s_2, C, D)$

$$= \underbrace{P(s_2, s_4, C)}_{=\frac{1}{8}} \cdot \underbrace{P(s_4, D)}_{=0} + \underbrace{P(s_2, s_5, C)}_{=\frac{6}{8}} \cdot \underbrace{P(s_5, D)}_{=1} + \underbrace{P(s_2, s_6, C)}_{=\frac{1}{8}} \cdot \underbrace{P(s_6, D)}_{-0} = \frac{3}{4}.$$

4.1.1 Quotient DTMC

Definition 4.6 (Quotient DTMC) *For WPE relation $\mathcal{R}$ on $\mathcal{D}$, the quotient DTMC $\mathcal{D}/_{\mathcal{R}}$ is defined by $\mathcal{D}/_{\mathcal{R}} = (S/_{\mathcal{R}}, P', AP, L', s'_0)$ where:*

- *$S/_{\mathcal{R}}$ is the set of all equivalence classes under $\mathcal{R}$,*
- *$P'(C, D) = wp(s', C, D)$ where $C, D \in S/_{\mathcal{R}}$ and $s' \in Pred(C)$,*
- *$L'(C) = L(s)$, where $s \in C$ and*
- *$s'_0 = C$ where $s_0 \in C = [s_0]_{\mathcal{R}}$.*

Note that $P'(C, D)$ is well-defined as by Def. 4.4 for any predecessors s', s'' of C it follows $wp(s', C, D) = wp(s'', C, D)$. Similarly, L' is well-defined as states in any equivalence class C are equally labeled.

Example 4.7 *For the DTMC $\mathcal{D}$ in Fig 4.2 (left), the quotient $\mathcal{D}/_{\mathcal{R}}$ under WPE $\mathcal{R}$ induced by the partitioning $\{\{s_0\},\{s_1,s_2,s_3\},\{s_4,s_5,s_6\},\{s_7\}\}$ is shown in Fig. 4.2 (right).*

Definition 4.8 *Any DTMC $\mathcal{D}$ and its quotient $\mathcal{D}/_{\mathcal{R}}$ under WPE relation $\mathcal{R}$ are $\hat{=}$-related, denoted by $\mathcal{D} \hat{=} \mathcal{D}/_{\mathcal{R}}$, if and only if there exists a WPE relation $\mathcal{R}^*$ defined on the disjoint union of state space $S \uplus S/_{\mathcal{R}}$ such that*

$$\forall C \in S/_{\mathcal{R}}, \forall s \in C \implies (s, C) \in \mathcal{R}^*$$

.

Remark 4.9 *Note that the probability matrix, say P'', on $S \uplus S/_{\mathcal{R}}$ is defined by: $P''(s,s') = P(s,s')$ if $s,s' \in S$, $P''(s,s') = P'(s,s')$ if $s,s' \in S/_{\mathcal{R}}$, and 0 otherwise.*

Next, we show that any DTMC $\mathcal{D}$ and its quotient under WPE relation are $\hat{=}$-related.

Theorem 4.10 *For any DTMC $\mathcal{D}$ and WPE $\mathcal{R}$ on S : $\mathcal{D} \hat{=} \mathcal{D}/_{\mathcal{R}}$.*

Note that union of WPEs is not necessarily a WPE. Intuitively it means that the original DTMC $\mathcal{D}$ can be minimized in different ways. This can be observed from the example given below.

Example 4.11 *Consider the DTMC $\mathcal{D}$ shown in Fig. 4.3. In this case DTMC $\mathcal{D}$ can be minimized in two different ways using WPEs as shown in Fig. 4.4. Here DTMC $\mathcal{D}'$ shown in Fig. 4.4 (left) is the quotient system for the WPE relation $\mathcal{R}$ induced by set $\{\{s_0\},\{s_1\},\{s_2\},\{s_3,s_4,s_5\},\{s_6,s_7,s_8\},\{s_9\},\{s_{10},s_{12}\},\{s_{14}\},\{s_{11},s_{13}\}\}$. DTMC $\mathcal{D}''$ shown in Fig. 4.4 (right) is the quotient system for the WPE $\mathcal{R}'$ induced by set $\{\{s_0\},\{s_1\},\{s_2\},\{s_3\},\{s_4,s_5,s_6,s_7\},\{s_8\},\{s_9\},\{s_{14}\},\{s_{10},s_{12}\},\{s_{11},s_{13}\}\}$. It is easy to check that $\mathcal{R} \cup \mathcal{R}'$ is not a WPE.*

Repeated minimization. Next, we show that WPE can be used for repeated minimization of a DTMC. Intuitively, this means that if a quotient system $\mathcal{D}'$ has been obtained from a DTMC $\mathcal{D}$ under WPE $\mathcal{R}$, then it might still be possible to further reduce $\mathcal{D}'$ to $\mathcal{D}''$ under some KME $\mathcal{R}'$. Consider the DTMC shown in Fig. 4.5 (left). DTMC in Fig. 4.5 (middle) is the quotient system for the WPE induced by the partition $\{\{s_0\},\{s_1,s_2\},\{s_3,s_4\},\{s_5\},\{s_6\},\{s_7\},\{s_8\}\}$. DTMC in Fig. 4.5 (right) is the quotient of the DTMC Fig. 4.5 (middle) for the

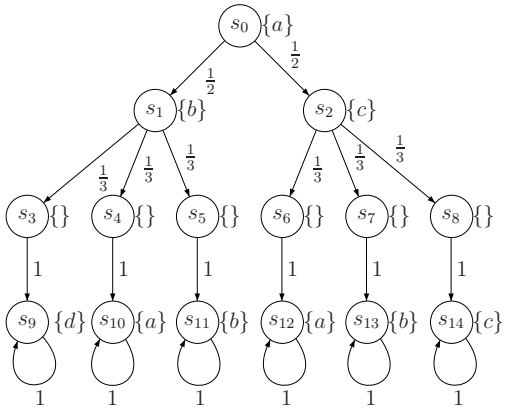

Figure 4.3: An example DTMC $\mathcal{D}$

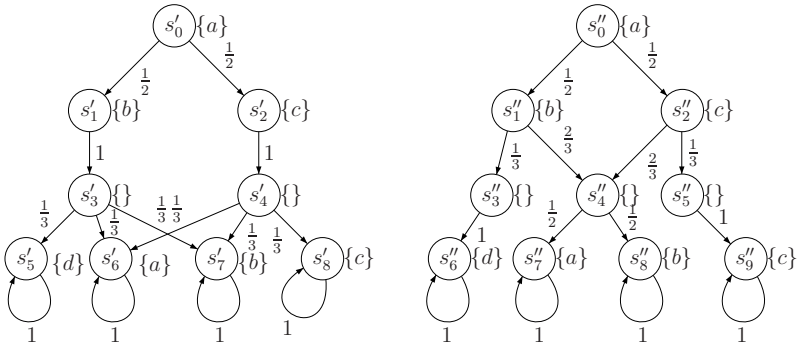

Figure 4.4: Union of WPEs is not necessarily a WPE

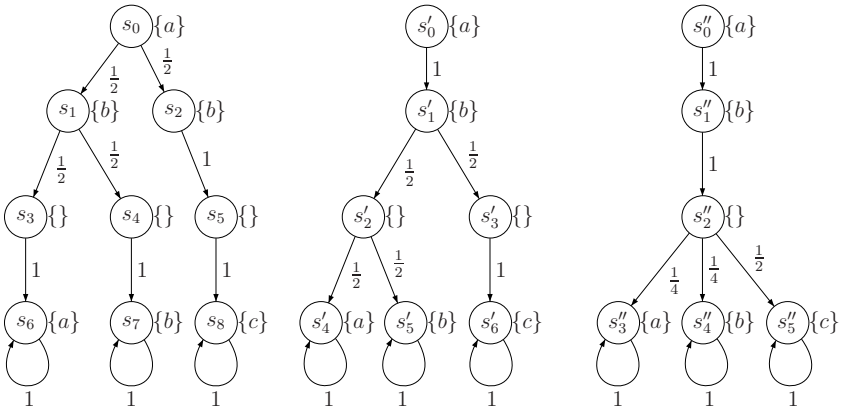

Figure 4.5: Repeated minimization

WPE induced by the parttion $\{\{s'_0\},\{s'_1\},\{s'_2,s'_3\},\{s'_4\},\{s'_5\},\{s'_6\}\}$. It is easy to check that s_3, s_4, s_5 in the original system cannot be merged in one shot, since s_1 can reach states labeled with atomic propositions a and b in two steps via s_3 and s_4 respectively, but s_2 cannot reach such states. This is no longer a problem once s_1 and s_2 are merged as shown in Fig. 4.5 (middle) as s'_2, s'_3 now have a single predecessor, i.e., s'_1. Next, we investigate the relationship of WPE to bisimulation.

4.1.2 WPE vs. Bisimulation

Definition 4.12 (Bisimulation [10, 104]) *Equivalence $\mathcal{R}$ on S is a probabilistic* bisimulation *on $\mathcal{D}$ if for any $(s_1, s_2) \in \mathcal{R}$ we have: $L(s_1) = L(s_2)$, and $P(s_1, C) = P(s_2, C)$ for all C in $S/_{\mathcal{R}}$. s_1 and s_2 are* bisimilar*, denoted $s_1 \sim s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$.*

These conditions require that any two bisimilar states are equally labeled and have identical cumulative probabilities to move to any equivalence class $C \in S/_{\mathcal{R}}$. Note that these conditions coincide with lumping.

Lemma 4.13 *$\sim$ is strictly finer than WPE.*

From [10], we know that $\sim$ coincides with probabilistic simulation equivalence for DTMCs. Therefore we get the following corollary:

Corollary 4.14 *Probabilistic simulation equivalence is strictly finer than WPE.*

Remark 4.15 *From Fig. 4.2 it can be observed that states s_5 and s_6 cannot be merged under $\sim$, as s_6 can move to s_0 but there is no direct successor s of s_5 with $s \sim s_0$ (Note that $L(s_0) \neq L(s_7)$). Similarly, s_2 and s_3 cannot be merged under $\sim$. This shows that $WPE \not\Rightarrow \sim$.*

4.1.3 Preservation of ω-Regular Properties

In this section we investigate the linear-time properties for DTMCs that are preserved by WPE. We study a more general class of linear-time properties, i.e., ω-regular properties that are defined over traces, i.e., infinite sequence of symbols. These include, e.g., properties of the form: every time the process tries to send a message, it eventually succeeds in sending it. Note that the preservation of ω-regular properties implies the preservation of LTL formulas and

transient-state probabilities [9]. These preservation results can be exploited for model checking, by reducing DTMC models under consideration prior to carrying out the verification, provided there is an algorithm. This may speed up the verification as (mostly) a smaller model needs to be checked.

In the context of branching-time equivalence relations, probabilistic bisimulation coincides with the logical equivalence of the branching-time logic's PCTL and PCTL* [4, 49, 73]. PCTL* and ω-regular properties have incomparable expressiveness [9].

Definition 4.16 (ω-regular language) *A language $\mathcal{L} \subseteq (2^{AP})^\omega$ is called ω-regular if $\mathcal{L} = \mathcal{L}_\omega(G)$ for some ω-regular expression G over 2^{AP}.*

For instance, the language consisting of all infinite words over $\{a, b\}$ that contain only finitely many a's is ω-regular since it is given by the ω-regular expression $(a+b)^*b^\omega$. ω-regular languages possess several closure properties: they are closed under union, intersection and complementation [150, pp. 61-77], [9, pp. 172-198].

Definition 4.17 (ω-regular property) *Linear-time property P over AP is called ω-regular if P is an ω-regular language over the alphabet 2^{AP}.*

Theorem 4.18 *[150, pp. 53-59] The class of languages accepted by DRAs agrees with the class of ω-regular languages.*

Intuitively, this theorem says that any property P that can be expressed using ω-regular language L is also expressible using some DRA $\mathcal{A}$ and vice versa. Next we show that the probability of satisfying a DRA specification for any DTMC coincides with the probability for its quotient under WPE.

Definition 4.19 (DRA) *A* deterministic Rabin automaton *(DRA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where:*

- *Q is a nonempty finite set of locations,*
- *Σ is a finite alphabet,*
- *$\delta : Q \times \Sigma \to Q$ is the transition function,*
- *q_0 is the initial location,*
- *$Acc \subseteq 2^Q \times 2^Q$ where:*
  the run $q_0 q_1 q_2 \ldots$ is accepting *if there exists a pair $(L_i, K_i) \in Acc$ such that*

$$(\exists n \geq 0. \forall m \geq n. q_m \notin L_i) \wedge (\exists^\infty n \geq 0. q_n \in K_i).$$

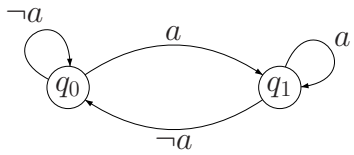

Figure 4.6: An example DRA

Intuitively a DRA is a finite-state automaton with the same components as non-deterministic Büchi automaton (NBA) [150, pp. 3-7], [9, pp. 173-178] except for the acceptance condition. The acceptance condition of a DRA is given by a set of pairs of states: $\{(L_i, K_i) | 0 < i \leq k\}$ with $L_i, K_i \subseteq Q$. A run of a DRA is accepting if for some pair (L_i, K_i) the states in L_i are visited finitely often and some states (at least one) in K_i infinitely often. A DRA is deterministic since it has a single initial state and the successor location of a transition is uniquely determined. The edge $q \xrightarrow{a} q'$ asserts that the DRA $\mathcal{A}$ moves from location q to q' when the input symbol is a. An infinite path of DRA $\mathcal{A}$ has the form $\rho = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \ldots$.

Example 4.20 *Consider the DRA $\mathcal{A}$ in Fig. 4.6. Let $AP = \{a\}$, $\Sigma = \{\{a\}, \varnothing\}$, $Q = \{q_0, q_1\}$, $Acc = \{(\{q_0\}, \{q_1\})\}$, and q_0 is the initial state. The runs accepted by $\mathcal{A}$ are those which eventually stay forever in state q_1. The ω-regular property expressed by this DRA is given by: eventually forever a ($\Diamond\Box a$).*

Before defining the *probability of paths in DTMC $\mathcal{D}$ that are accepted by DRA $\mathcal{A}$*, we first define two auxiliary concepts namely *product Markov chain ($\mathcal{D} \otimes \mathcal{A}$)* and *accepting BSCCs in $\mathcal{D} \otimes \mathcal{A}$*.

Definition 4.21 (Product Markov chain) *[9, pp. 802-803] Let $\mathcal{D} = (S, P, AP, L, s_0)$ be a DTMC and $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$ be a DRA. The product $\mathcal{D} \otimes \mathcal{A}$ is the Markov chain:*

$$\mathcal{D} \otimes \mathcal{A} = (S \times Q, P', \langle s_0, q \rangle, AP', L')$$

where L_i, K_i serve as atomic propositions in $\mathcal{D} \otimes \mathcal{A}$ if the acceptance condition of $\mathcal{A}$ is $Acc = \{(L_1, K_1), \ldots, (L_k, K_k)\}$. The set of these atomic propositions is AP'. The labeling function L' in $\mathcal{D} \otimes \mathcal{A}$ is: if $H \in \{L_1, \ldots, L_k, K_1 \ldots, K_k\}$, then $H \in L'(\langle s, q \rangle)$ if and only if $q \in H$. The initial state of $\mathcal{D} \otimes \mathcal{A}$, i.e., $\langle s_0, q \rangle$ is s.t. $q = \delta(q_0, L(s_0))$. The transition probabilities in $\mathcal{D} \otimes \mathcal{A}$ are given by:

$$P'(\langle s, q \rangle, \langle s', q' \rangle) = \begin{cases} P(s, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise.} \end{cases}$$

The product Markov chain is intuitively the synchronous product of DTMC $\mathcal{D}$ and DRA $\mathcal{A}$ s.t. transition $s \to s'$ in $\mathcal{D}$ is matched with edge $q \xrightarrow{L(s)} q'$.

Definition 4.22 (Accepting BSCCs in $\mathcal{D} \otimes \mathcal{A}$) *A BSCC T in $\mathcal{D} \otimes \mathcal{A}$ is accepting if and only if there exists some index $i \in \{1, \ldots, k\}$ such that*

$$T \cap (S \times L_i) = \varnothing \text{ and } T \cap (S \times K_i) \neq \varnothing.$$

Let us formally define the paths of DTMC $\mathcal{D}$ that are accepted by DRA $\mathcal{A}$.

Definition 4.23 (DTMC paths accepted by a DRA) *Let DTMC $\mathcal{D} = (S, P, AP, L, s_0)$ and DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$. The DTMC path $\pi = s_0 \to s_1 \to s_2 \ldots$ is accepted by $\mathcal{A}$ if there exists a corresponding DRA path*

$$q_0 \xrightarrow{L(s_0)} q_1 \xrightarrow{L(s_1)} q_2 \ldots$$

such that for path $\langle s_0, q_1\rangle\langle s_1, q_2\rangle \ldots$ in $\mathcal{D} \otimes \mathcal{A}$, $\langle s_i, q_{i+1}\rangle \in T$ for some $i \geq 0$, where T is an accepting BSCC in $\mathcal{D} \otimes \mathcal{A}$.

Since the product Markov chain is also a DTMC it will eventually reach a BSCC and visits all its states infinitely often. Let $Paths^{\mathcal{D}}(\mathcal{A}) = \{\pi \in Paths^{\mathcal{D}} | \pi \text{ is accepted by } \mathcal{A}\}$. Note that for any DTMC $\mathcal{D}$ and DRA $\mathcal{A}$, the set $Paths^{\mathcal{D}}(\mathcal{A})$ is measurable [9, pp. 804-805].

Definition 4.24 *For DTMC $\mathcal{D}$ and DRA $\mathcal{A}$, let* $\Pr(\mathcal{D} \models \mathcal{A}) = \Pr\left(Paths^{\mathcal{D}}(\mathcal{A})\right)$.

Stated in words, $\Pr(\mathcal{D} \models \mathcal{A})$ denotes the probability of all the paths in DTMC $\mathcal{D}$ that are accepted by DRA $\mathcal{A}$.

Theorem 4.25 (Preservation of DRA specifications) *For any DTMC $\mathcal{D}$, a WPE $\mathcal{R}$ on $\mathcal{D}$ and DRA $\mathcal{A}$:*

$$\Pr(\mathcal{D} \models \mathcal{A}) \;=\; \Pr(\mathcal{D}/_{\mathcal{R}} \models \mathcal{A}).$$

Intuitively this theorem says that the probability of all the paths in Markov chain $\mathcal{D}$ satisfying DRA $\mathcal{A}$ equals the probability of all the paths in $\mathcal{D}/_{\mathcal{R}}$ that satisfy $\mathcal{A}$.

For a DTMC $\mathcal{D}$, two types of state probabilities are distinguished: transient-state probabilities where the system is considered at a given discrete time step n, and steady-state probabilities where the system is considered "on the long run", i.e., when an equilibrium has been reached.

Definition 4.26 (Transient-state probability) *The probability of being in state s' at time instant $n \in \mathbb{N}$ starting from s is defined as follows:*

$$\mathcal{T}(s, s', n) = Pr_s\{\pi \in Paths(s) | \pi[n] = s'\}$$

Definition 4.27 (Steady-state probability) *The probability of being in state s' in the long run starting from s is defined as follows:*

$$\mathcal{S}(s, s') = \lim_{n \to \infty} Pr_s\{\pi \in Paths(s) | \pi@n = s'\}.$$

Note that we only compute steady-state probability on DTMCs for which the limit does exist. For ergodic DTMCs the initial state does not have any influence on the value of $\mathcal{S}(s, s')$, but we keep this notation as in the case of reducible DTMCs the initial state has influence on this value.

Corollary 4.28 *WPE preserves transient-state probabilities.*

In Section 4.3, we also show that WPE preserves steady-state probabilities.

4.2 Synchronous Parallel Composition

In this section, we show that WPE is compositional w.r.t. synchronous parallel composition of DTMCs [151, 152]. This result is useful for analyzing synchronous distributed algorithms and synchronous hardware circuits where processes progress in a lock-step fashion. For example we want to compose a DTMC $\mathcal{D}_1$ with DTMC $\mathcal{D}_2$ and these DTMCs have n and m states respectively. Then the resulting DTMC $\mathcal{D}_1 \otimes \mathcal{D}_2$ can have $n \cdot m$ states (in the worst case) so it is worthwhile to replace this composition by a smaller DTMC $\mathcal{D}' \stackrel{\circ}{=}$-related to $\mathcal{D}_1$. An interesting and practically relevant case study of failure behavior of Negated AND (NAND) multiplexing has been investigated in [121]. In that paper, the authors construct a PRISM module for each of the N NAND gates in the stage, and then combining these modules through synchronous parallel composition. Synchronous parallel composition operator ($\otimes$) for DTMCs is formally defined as:

Definition 4.29 (Synchronous parallel composition [151, 152]) *Let $\mathcal{D}_1 = (S_1, P_1, AP_1, L_1, s_{01})$ and $\mathcal{D}_2 = (S_2, P_2, AP_2, L_2, s_{02})$ be two DTMCs. We say that $s \xrightarrow{p_i}_i s'$ if $p_i = P_i(s, s') > 0$ for $i = 1, 2$. The* synchronous parallel composition

of $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *is* $\mathcal{D}_1 \otimes \mathcal{D}_2 = (S_1 \times S_2, P, AP_1 \cup AP_2, L, (s_{01}, s_{02}))$*, where* $L((s_1, s_2)) = L(s_1) \cup L(s_2)$*, and* P *is given as follows:*

$$\frac{s_1 \xrightarrow{p_1}_1 s_1' \wedge s_2 \xrightarrow{p_2}_2 s_2'}{(s_1, s_2) \xrightarrow{p_1 \cdot p_2} (s_1', s_2')}.$$

Intuitively, both Markov chains proceed in a lock-step fashion. Transition probabilities thus result in the product of individual transition probabilities. Note that $\otimes$ is commutative and associative.

Theorem 4.30 *Let* $\mathcal{D}$ *be a DTMC and* $\mathcal{R}$ *a WPE on* $\mathcal{D}$*. Then for any DTMC* $\mathcal{D}_1$*:*

$$(\mathcal{D} \otimes \mathcal{D}_1) \stackrel{\circ}{=} (\mathcal{D}/_{\mathcal{R}} \otimes \mathcal{D}_1).$$

4.3 Reward Properties

Rewards are costs or bonuses associated to states in S. Discrete-time Markov reward models are useful for analyzing for instance the average behavior of executions in DTMCs. For example in a battery-powered embedded system, a measure of interest is the expected power consumption during operation. Another example is a communication system where a sender and a receiver can transfer messages via an unreliable channel, in this case an interesting measure of interest is the expected number of attempts to send a message until correct delivery. An interesting case study where the behavior of IPv4 zeroconf protocol has been modeled as an DMRM is presented in [2, 22]. A discrete-time Markov reward model is a DTMC where the states are augmented with rewards, which are non-negative real-valued numbers. The reward associated with a state s is earned when s is left. More formally, DMRM $\mathcal{MD}$ is defined as follows:

Definition 4.31 ([2, 9]) *A discrete-time Markov* reward *model (DMRM)* $\mathcal{MD}$ *is a pair* $(\mathcal{D}, rew)$ *with DTMC* $\mathcal{D} = (S, P, AP, L, s_0)$ *and* $rew : S \to \mathbb{R}_{\geq 0}$ *a reward assignment function that associates a non-negative real reward to any state in* S*. Real number* $rew(s)$ *denote the reward earned on leaving state* s*.*

The cumulative reward earned along a finite path π of length n is defined as $rew(\pi) = \sum_{i=0}^{n-1} rew(s_i)$. As rewards are earned on leaving a state, $rew(s_n)$ is not considered in the cumulative reward of π.

Example 4.32 *Consider the DTMC $\mathcal{D}$ in Fig. 4.2 (left) with reward structure rew defined by $rew(s_0) = 1$, $rew(s_1) = rew(s_2) = rew(s_3) = 2$, $rew(s_4) = rew(s_5) = rew(s_6) = 3$ and $rew(s_7) = 0$. Then the cumulative reward earned along the finite path $\pi = s_0 \to s_1 \to s_5 \to s_7$ is given by $rew(s_0) + rew(s_1) + rew(s_5) = 6$.*

Reward-extended LTL (RLTL)

Next we define the syntax and semantics of reward-extended LTL (RLTL).

Definition 4.33 (Syntax of RLTL)

$$\varphi \ ::= \ tt \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc_J \varphi \mid \varphi \,\mathrm{U}_J\, \varphi$$

where $J \subseteq \mathbb{R}_{\geq 0}$ is a nonempty interval with rational bounds, and $a \in AP$.

Definition 4.34 (Satisfaction relation for RLTL) *Given an infinite path π in DMRM $\mathcal{MD}$, the satisfaction relation for RLTL, denoted by $\models$, is defined inductively by:*

$$\begin{array}{lcll}
\pi & \models & tt & \\
\pi & \models & a & \textit{iff} \quad a \in L(\pi[0]) \\
\pi & \models & \neg\varphi & \textit{iff} \quad \textit{not } \pi \models \varphi \\
\pi & \models & \varphi_1 \wedge \varphi_2 & \textit{iff} \quad \pi \models \varphi_1 \textit{ and } \pi \models \varphi_2 \\
\pi & \models & \bigcirc_J \varphi & \textit{iff} \quad \pi[1...] \models \varphi \wedge rew(\pi[0]) \in J \\
\pi & \models & \varphi_1 \,\mathrm{U}_J\, \varphi_2 & \textit{iff} \quad \exists j \geq 0.\, (\pi[j...] \models \varphi_2 \wedge \\
 & & & \quad \forall 0 \leq i < j.\, \pi[i...] \models \varphi_1 \wedge \sum_{i=0}^{j-1} rew(\pi[i]) \in J).
\end{array}$$

The semantics for the propositional fragment is straightforward. Path π satisfies $\bigcirc_J\varphi$ if the path starting from the next state satisfies φ and the reward earned along π up to the current state belongs to the interval J. Path π satisfies $\varphi_1 \,\mathrm{U}_J\, \varphi_2$ whenever it satisfies $\varphi_1 \,\mathrm{U}\, \varphi_2$ and the cumulative reward earned along π upto reaching φ_2 belongs to the interval J. The standard temporal operators like $\Diamond$ ("eventually") and its reward based variant $\Diamond_J$ are derived in the following way: $\Diamond_J\varphi = tt \,\mathrm{U}_J\, \varphi$. Similarly, $\Box$ ("globally") and its reward based variant are derived as follows:

$$\Box_J\varphi = \neg(\Diamond_J\, \neg\varphi).$$

Example 4.35 *RLTL can be used to specify various interesting properties:*

- $\Box(down \rightarrow \Diamond_{[0,5]} up)$, *which asserts that whenever the system is down, it should be eventually up again while accumulating a reward between 0 and 5.*
- $\Box(down \rightarrow alarm\, \mathrm{U}_{[5,15]}\, up)$, *which asserts that whenever the system is down, an alarm should ring until it is up again and the reward accumulated should be between 5 and 15.*

Definition 4.36 (Probability of RLTL formulas) *The probability that state s of $\mathcal{MD}$ satisfies RLTL formula φ refers to the probability for the sets of paths for which that formula holds:*

$$\Pr(s \models \varphi) \;=\; \Pr_{s}(\pi \in Paths(s) \mid \pi \models \varphi).$$

Definition 4.37 (WPE for DMRM) *To accommodate the notion of rewards in the definition of WPE we extend the conditions in Def. 4.4 as follows:*

$$\forall (s_1, s_2) \in \mathcal{R} \text{ it holds} : rew(s_1) = rew(s_2).$$

Note that in the quotient system for any $C \in S/_{\mathcal{R}}$, $rew(C) = rew(s)$ where $s \in C$.

Definition 4.38 (Expected reward for RLTL formulas) *For state s and RLTL formula φ, the expected reward for $s \models \varphi$ is defined as follows:*

$$\mathcal{E}(s \models \varphi) = \sum_{r=0}^{\infty} r \cdot \Pr_{s}\{\pi \in Paths(s) \mid \pi \models \varphi \wedge rew(\pi) = r\}.$$

Example 4.39 *Consider the DTMC $\mathcal{D}$ in Fig. 4.2 (left). Let the labeling function and reward structure for $\mathcal{D}$ is same as in Example 2.12 and Example 4.32 respectively. Let $\varphi = ((c \vee b)\mathrm{U}_{[0,5]}a)$, then $\mathcal{E}(\varphi)$ is as follows:*

$$\begin{aligned}
\mathcal{E}(\varphi) &= Pr_{s_0}(Cyl(s_0, s_1, s_4)) \cdot rew(s_0 \rightarrow s_1 \rightarrow s_4) \\
&+ Pr_{s_0}(Cyl(s_0, s_1, s_5)) \cdot rew(s_0 \rightarrow s_1 \rightarrow s_5) \\
&+ Pr_{s_0}(Cyl(s_0, s_2, s_5)) \cdot rew(s_0 \rightarrow s_2 \rightarrow s_5) \\
&+ Pr_{s_0}(Cyl(s_0, s_2, s_6)) \cdot rew(s_0 \rightarrow s_2 \rightarrow s_6) \\
&+ Pr_{s_0}(Cyl(s_0, s_2, s_4)) \cdot rew(s_0 \rightarrow s_2 \rightarrow s_4) \\
&+ Pr_{s_0}(Cyl(s_0, s_3, s_5)) \cdot rew(s_0 \rightarrow s_3 \rightarrow s_5) \\
&+ Pr_{s_0}(Cyl(s_0, s_3, s_6)) \cdot rew(s_0 \rightarrow s_3 \rightarrow s_6) = \frac{123}{56}
\end{aligned}$$

.

Theorem 4.40 *Let $\mathcal{MD}$ be a DMRM and $\mathcal{R}$ be a WPE on $\mathcal{MD}$. Then for any RLTL formula φ:*

$$\mathcal{E}(\mathcal{MD} \models \varphi) = \mathcal{E}(\mathcal{MD}/_R \models \varphi).$$

Definition 4.41 (Instantaneous reward) *The expected instantaneous reward earned by being in state s' at time instant n starting from state s is defined as follows:*

$$irew(s, s', n) = \mathcal{T}(s, s', n) \cdot rew(s').$$

Theorem 4.42 *Let $\mathcal{MD}$ be a DMRM and $\mathcal{R}$ be a WPE on $\mathcal{MD}$. Then we have:*

$$\sum_{s' \in C} irew(s_0, s', n) = irew(s'_0, C, n),$$

where $C \in S/_{\mathcal{R}}$ and s'_0 is the initial state of $\mathcal{MD}/_{\mathcal{R}}$.

Definition 4.43 (Long-run reward) *The expected long-run reward earned by being in state s' starting from s is defined as follows:*

$$lrew(s, s') = \mathcal{S}(s, s') \cdot rew(s').$$

Theorem 4.44 *Let $\mathcal{MD}$ be a DMRM and $\mathcal{R}$ be a WPE on $\mathcal{MD}$. Then we have:*

$$\sum_{s' \in C} lrew(s_0, s') = lrew(s'_0, C),$$

where $C \in S/_{\mathcal{R}}$ and s'_0 is the initial state of $\mathcal{MD}/_{\mathcal{R}}$.

Multiple rewards. Note that the logic RLTL can be easily extended such that properties over models equipped with multiple reward structures can be expressed [2]. The preservation results presented in this section also carry over to multiple reward models.

4.4 Related Work

Various branching-time relations on DTMCs have been defined such as weak and strong variants of bisimulation equivalence and simulation pre-orders [29, 66, 86, 87, 88, 93, 104]. Their compatibility to (fragments of) probabilistic variants of Computation Tree Logic (PCTL) [73] has been thoroughly investigated [10]. More specifically, strong bisimulation for DTMCs coincides with equivalence

for PCTL. Similarly, weak bisimulation for DTMCs coincides with equivalence for PCTL without next operator. Note that strong bisimulation coincides with strong simulation equivalence for DTMCs. In the weak setting, weak bisimulation coincides with weak simulation equivalence for DTMCs. In addition, it has been proved that strong simulation and weak simulation for DTMCs, preserve a safe fragment of PCTL and PCTL without next operator, respectively [10].

Probabilistic model checking tools such as Probabilistic Symbolic Model Checker (PRISM) [98] and Markov Reward Model Checker (MRMC) [89] have been successfully used to model check PCTL properties on DTMCs. MRMC supports bisimulation based minimization for DTMCs.

In the linear-time setting, probabilistic trace equivalences [80, 139] and probabilistic testing equivalence [33, 37, 140, 141] have been defined for discrete-time or time-abstract probabilistic models. For the continuous case, Markovian testing equivalence has been proposed in [16]. In [157] the Markovian variants of several linear-time equivalences have been extensively investigated.

4.5 Conclusions

This chapter defines an equivalence relation (what we refer to as) weighted probabilistic equivalence (WPE) on DTMCs. The main contributions of this paper are as follows:

- We show that ω-regular properties specified on DTMCs are preserved under WPE quotienting.
- Next, we show that WPE is compositional w.r.t. synchronous parallel composition.
- Finally, these preservation results are extended to Markov reward models.

Chapter 5

Weighted Lumpability

This chapter focuses on a notion of lumpability that allows for a more aggressive state-space aggregation than ordinary lumpability for CTMCs [146].

As in the case of WPE, weighted lumpability (WL, for short) also considers a *two-step* perspective. Before explaining the main principle of WL, let us recall that every transition of a CTMC is labeled with a positive real number λ. This parameter indicates the rate of the exponential distribution, i.e., the probability of a λ-labeled transition to be enabled within t time units equals $1 - e^{-\lambda \cdot t}$. In fact, the average residence time in a state is determined as the reciprocal of the sum of the rates of its outgoing transitions. Roughly speaking, two states s and s' are weighted lumpable if for each pair of their direct predecessors the weighted rate to directly move to any equivalence class via the equivalence class $[s] = [s']$ coincides. The main principle is captured in Fig. 5.1 where $\lambda_{1,1} + \lambda_{1,2} = \lambda_{2,1} + \lambda_{2,2}$, and $\lambda_{C_1} = p_1 \cdot \lambda_{1,1}$, $\lambda_{C_2} = p_1 \cdot \lambda_{1,2} + p_2 \cdot \lambda_{2,1}$, $\lambda_{C_3} = p_2 \cdot \lambda_{2,2}$ with $p_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2}$ and $p_2 = \frac{\lambda_2}{\lambda_1 + \lambda_2}$. Here states s_1 and s_2 are weighted lumpable, as the probability to move from s_0 to all the states in the equivalence class C_i (for i=1, 2, 3) via all the states in $[s_1]$ is equal. This allows for the aggregation of s_1 and s_2, cf. the right CTMC in Fig. 5.1.

We define WL as a *structural* notion on CTMCs. We define the quotient under WL, and show that any CTMC is related to its quotient under WL. Our structural definition allows for a simple proof that WL is (strictly) coarser than bisimulation, i.e., ordinary lumpability. Our main focus and motivation, however, is to investigate the preservation of *linear real-time objectives* under WL. We first show that the probability of satisfying a deterministic timed automaton (DTA) [1] specification for any CTMC coincides with that probability for its quotient. This allows for an a priori state-space reduction in linear real-time CTMC

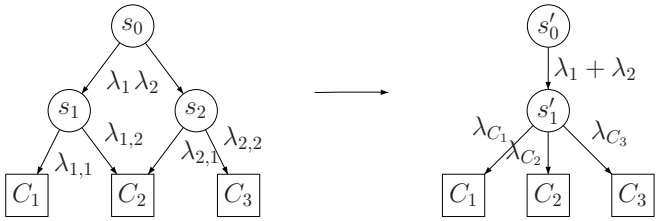

Figure 5.1: CTMC aggregation under weighted lumpability

model checking [12, 31], and implies the preservation of "flat" (i.e., unnested) timed reachability properties and CSL^{TA} formulas [50]. In addition, we study metric temporal logic (MTL) [95], a real-time variant of LTL that is typically used for timed automata (and not for CTMCs). DTA and MTL have incomparable expressiveness [9, 24, 109]. It is shown that WL-quotienting of CTMCs preserves the probability to satisfy any MTL formula. As a prerequisite result, we show that MTL formulas (interpreted on CTMCs) are measurable. We also discuss some case studies showing that WL can substantially reduce the size of the CTMC state space.

Organisation of this chapter. Section 5.1 defines weighted lumpability, treats some basic properties and discusses the preservation of linear real-time properties. Section 5.2 presents some case studies. Section 5.3 discusses related work. Finally, Section 5.4 concludes the chapter.

5.1 Weighted Lumpability

Before defining weighted lumpability, we first define some auxiliary concepts. All definitions in this section are relative to a CTMC $\mathcal{C} = (S, R, AP, L, s_0)$.

Definition 5.1 (Weighted rate) *For $s \in S$, and $C, D \subseteq S$, the function $wr : S \times 2^S \times 2^S \to \mathbb{R}_{\geq 0}$ is defined by:*

$$wr(s, C, D) = \sum_{s' \in C} P(s, s', C) \cdot R(s', D)$$

where $R(s', D) = \sum_{s'' \in D} R(s', s'')$ and $P(s, s', C)$ is defined as before (Def. 4.1).

Intuitively, $wr(s, C, D)$ is the (weighted) rate to move from s to some states in D in two steps via states of C. Since $P(s', D) = \frac{R(s', D)}{E(s')}$, $wr(s, C, D)$ equals $\sum_{s' \in C} P(s, s', C) \cdot P(s', D) \cdot E(s')$.

Example 5.2 *Consider the example in Fig. 5.2(a). Let* $C = \{s_3, s_4, s_5\}$. *Then* $P(s_1, s_3, C) = 1/4$, $P(s_1, s_4, C) = 3/4$, $P(s_2, s_4, C) = 3/4$, *and* $P(s_2, s_5, C) = 1/4$.

Example 5.3 *Consider the CTMC in Fig. 5.2(a). Let* $D = \{s_6\}$. *Then* $wr(s_1, C, D)$

$$= \underbrace{P(s_1, s_3, C)}_{=\frac{1}{4}} \cdot \underbrace{R(s_3, D)}_{=2} + \underbrace{P(s_1, s_4, C)}_{=\frac{3}{4}} \cdot \underbrace{R(s_4, D)}_{=0} = \frac{1}{2}.$$

and $wr(s_2, C, D)$

$$= \underbrace{P(s_2, s_4, C)}_{=\frac{3}{4}} \cdot \underbrace{R(s_4, D)}_{=0} + \underbrace{P(s_2, s_5, C)}_{=\frac{1}{4}} \cdot \underbrace{R(s_5, D)}_{=2} = \frac{1}{2}.$$

Similarly, for $D = \{s_7\}$, *we get* $wr(s_1, C, D)$

$$= \underbrace{P(s_1, s_3, C)}_{=\frac{1}{4}} \cdot \underbrace{R(s_3, D)}_{=0} + \underbrace{P(s_1, s_4, C)}_{=\frac{3}{4}} \cdot \underbrace{R(s_4, D)}_{=2} = \frac{3}{2}.$$

and $wr(s_2, C, D)$

$$= \underbrace{P(s_2, s_4, C)}_{=\frac{3}{4}} \cdot \underbrace{R(s_4, D)}_{=2} + \underbrace{P(s_2, s_5, C)}_{=\frac{1}{4}} \cdot \underbrace{R(s_5, D)}_{=0} = \frac{3}{2}.$$

The above ingredients allow for the following definition of weighted lumpability, the central notion in this chapter.

Definition 5.4 (WL) *Equivalence* $\mathcal{R}$ *on* S *is a* weighted lumping *(WL) on* $\mathcal{C}$ *if we have:*

1. $\forall (s_1, s_2) \in \mathcal{R}$ *it holds:* $L(s_1) = L(s_2)$ *and* $E(s_1) = E(s_2)$, *and*
2. $\forall C, D \in S/_{\mathcal{R}}$ *and* $\forall s', s'' \in Pred(C)$ *it holds:* $wr(s', C, D) = wr(s'', C, D)$.

States s_1, s_2 *are* weighted lumpable, *denoted by* $s_1 \cong s_2$, *if* $(s_1, s_2) \in \mathcal{R}$ *for some WL* $\mathcal{R}$.

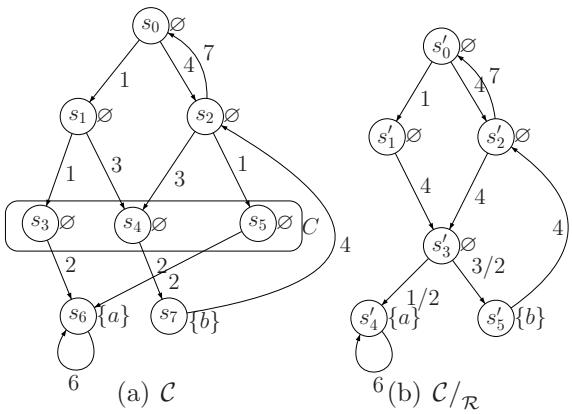

Figure 5.2: (a) A CTMC and (b) its quotient under weighted lumpability.

The first condition asserts that s_1 and s_2 are equally labeled and have identical exit rates. The second condition requires that for any two equivalence classes $C, D \in S/_{\mathcal{R}}$, where $S/_{\mathcal{R}}$ denotes the set consisting of all $\mathcal{R}$-equivalence classes, the weighted rate of going from any two predecessors of C to D via any state in C must be equal. Note that, by definition, any WL is an equivalence relation. Weighted lumpability coincides with Bernardo's notion of T-lumpability [16, 17] that is defined in an axiomatic manner for action-labeled CTMCs. Roughly speaking, two states are T-lumpable if their expected delays w.r.t. to any test process, put in parallel to the CTMC, coincide for both the states.

Example 5.5 *For the CTMC in Fig. 5.2(a), the equivalence relation induced by the partitioning* $\{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3, s_4, s_5\}, \{s_6\}, \{s_7\}\}$ *is a WL relation.*

5.1.1 Quotient CTMC

Definition 5.6 (Quotient CTMC) *For WL relation* $\mathcal{R}$ *on CTMC* $\mathcal{C}$*, the quotient CTMC* $\mathcal{C}/_{\mathcal{R}}$ *is defined by* $\mathcal{C}/_{\mathcal{R}} = (S/_{\mathcal{R}}, R', AP, L', s'_0)$ *where:*

- $S/_{\mathcal{R}}$ *is the set of all equivalence classes under* $\mathcal{R}$*,*
- $R'(C, D) = wr(s', C, D)$ *where* $C, D \in S/_{\mathcal{R}}$ *and* $s' \in Pred(C)$*,*
- $L'(C) = L(s)$*, where* $s \in C$ *and*
- $s'_0 = C$ *where* $s_0 \in C = [s_0]_{\mathcal{R}}$*.*

Note that $R'(C, D)$ is well-defined as for any predecessors s', s'' of C it follows $wr(s', C, D) = wr(s'', C, D)$. Similarly, L' is well-defined as states in any equivalence class C are equally labeled.

Example 5.7 *For the CTMC $\mathcal{C}$ in Fig. 5.2(a), the quotient $\mathcal{C}/_{\mathcal{R}}$ under the WL $\mathcal{R}$ induced by the partitioning $\{\{s_0\},\{s_1\},\{s_2\},\{s_3,s_4,s_5\},\{s_6\},\{s_7\}\}$ is shown in Fig. 5.2(b).*

Next, we show that any CTMC $\mathcal{C}$ and its quotient under WL relation are $\cong$-related.

Definition 5.8 *Any CTMC $\mathcal{C}$ and its quotient $\mathcal{C}/_{\mathcal{R}}$ under WL $\mathcal{R}$ are $\cong$-related, denoted by $\mathcal{C} \cong \mathcal{C}/_{\mathcal{R}}$, if and only if there exists a WL relation $\mathcal{R}^*$ defined on the disjoint union of state space $S \uplus S/_{\mathcal{R}}$ such that*

$$\forall C \in S/_{\mathcal{R}}, \forall s \in C \implies (s, C) \in \mathcal{R}^*$$

.

Theorem 5.9 *For any CTMC $\mathcal{C}$ and WL $\mathcal{R}$ on $S : \mathcal{C} \cong \mathcal{C}/_{\mathcal{R}}$.*

Note that union of WL relations is not necessarily a WL relation. Intuitively it means that the original CTMC $\mathcal{C}$ can be reduced in different ways.

Example 5.10 *Consider the CTMC $\mathcal{C}$ shown in Fig. 5.3. In this case CTMC $\mathcal{C}$ can be minimized in two different ways as shown in Fig. 5.4. CTMC $\mathcal{C}'$ shown in Fig. 5.4 (left) is the quotient system for the WL relation $\mathcal{R}$ induced by set $\{\{s_0\},\{s_1\},\{s_2\},\{s_3,s_4,s_5\},\{s_6,s_7,s_8\},\{s_9\},\{s_{14}\},\{s_{10},s_{12}\},\{s_{11},s_{13}\}\}$. CTMC $\mathcal{C}''$ shown in Fig. 5.4 (right) is the quotient system for the WL $\mathcal{R}'$ induced by set $\{\{s_0\},\{s_1\},\{s_2\},\{s_3\},\{s_4,s_5,s_6,s_7\},\{s_8\},\{s_9\},\{s_{14}\},\{s_{10},s_{12}\},\{s_{11},s_{13}\}\}$. It is easy to check that $\mathcal{R} \cup \mathcal{R}'$ is not a WL relation.*

Repeated minimization. Next, we show that WL can be used for repeated minimization of a CTMC. Intuitively, this means that if a quotient system $\mathcal{C}'$ has been obtained from a CTMC $\mathcal{C}$ under WL $\mathcal{R}$, then it might still be possible to further reduce $\mathcal{C}'$ to $\mathcal{C}''$ under some WL equivalence $\mathcal{R}'$. Consider the CTMC shown in Fig. 5.5 (left). CTMC in Fig. 5.5 (middle) is the quotient system for the WL induced by the partition $\{\{s_0\},\{s_1,s_2\},\{s_3,s_4\},\{s_5\},\{s_6\},\{s_7\},\{s_8\}\}$. CTMC in Fig. 5.5 (right) is the quotient of the CTMC Fig. 5.5 (middle) for the WL induced by the parttion $\{\{s'_0\},\{s'_1\},\{s'_2,s'_3\},\{s'_4\},\{s'_5\},\{s'_6\}\}$. It is easy to check that s_3, s_4, s_5 in the original system cannot be merged in one shot, since s_1 can reach states labeled with atomic propositions a and b in two steps via s_3 and s_4 respectively, but s_2 cannot reach such states. This is no longer a problem once s_1 and s_2 are merged as shown in Fig. 5.5 (middle) as s'_2, s'_3 now have a single predecessor, i.e., s'_1.

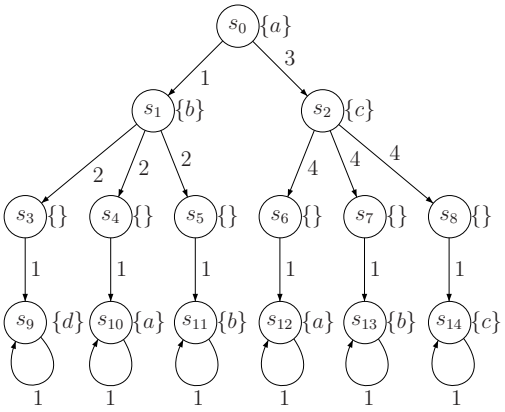

Figure 5.3: An example CTMC $\mathcal{C}$

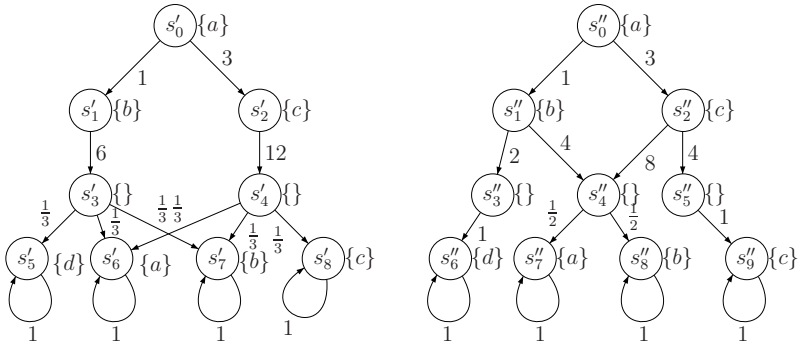

Figure 5.4: Union of WL relations is not necessarily a WL relation

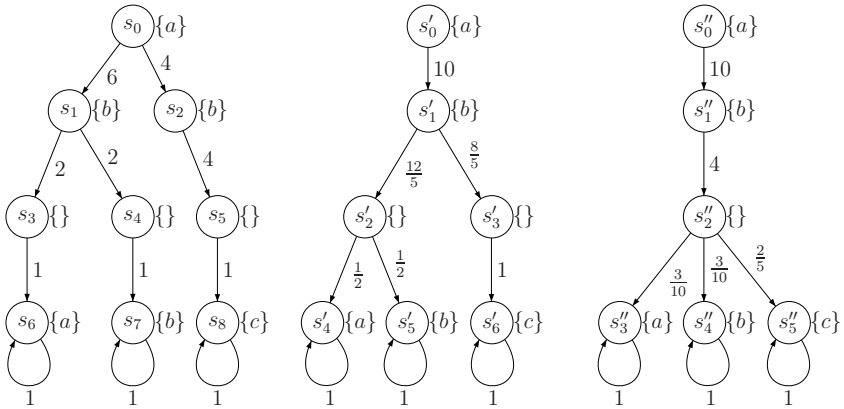

Figure 5.5: Repeated minimization

5.1.2 WL vs. Bisimulation

Next, we investigate the relationship of WL to bisimulation, i.e., ordinary lumping [10, 29]. This relationship is not novel; it is also given for T-lumpability in [16], but its proof is now quite simple thanks to the simplicity of the definition of WL.

Definition 5.11 (Bisimulation [10]) *Equivalence $\mathcal{R}$ on S is a stochastic bisimulation on $\mathcal{C}$ if for any $(s_1, s_2) \in \mathcal{R}$ we have: $L(s_1) = L(s_2)$, and $R(s_1, C) = R(s_2, C)$ for all C in $S/_{\mathcal{R}}$. s_1 and s_2 are bisimilar, denoted $s_1 \sim s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$.*

These conditions require that any two bisimilar states are equally labeled and have identical cumulative rates to move to any equivalence class C. Note that as $R(s, C) = P(s, C) \cdot E(s)$, the condition on the cumulative rates can be reformulated as $P(s_1, C) = P(s_2, C)$ for all $C \in S/_{\mathcal{R}}$ and $E(s_1) = E(s_2)$.

Lemma 5.12 *$\sim$ is strictly finer than WL.*

This lemma says that state space reduction under WL can potentially be larger than for strong stochastic bisimilarity. Consider the equivalence class $C = \{s_3, s_4, s_5\}$ in Fig. 5.2 (left). Here s_3, s_4 are WL related, but $s_3 \not\sim s_4$ since s_3 can reach an a-state while s_4 cannot. From [10], we know that $\sim$ coincides with stochastic simulation equivalence for CTMCs. Therefore we get the following corollary:

Corollary 5.13 *Stochastic simulation equivalence is strictly finer than WL.*

5.1.3 Preservation of DTA Specifications

Bisimulation equivalence coincides with the logical equivalence of the branching-time logic CSL [10], a probabilistic real-time variant of CTL [11]. This implies that bisimilar states satisfy the same CSL formulas, a property that—thanks to efficient minimisation algorithms [48]— is exploited by model checkers to minimise the state space prior to verification. In order to investigate the kind of real-time properties for CTMCs that are preserved by WL, we study in this section *linear* real-time objectives that are given by Deterministic Timed Automata (DTA) [1]. These include, e.g., properties of the form: what is the probability to reach a given target state within the deadline, while avoiding "forbidden" states and not

staying too long in any of the "dangerous" states on the way. Such properties can neither be expressed in CSL nor in dialects thereof [50]. A model-checking algorithm that verifies a CTMC against a DTA specification has recently been developed [31]; first experimental results are provided in [12]. The key issue is to compute the probability of all CTMC paths that are accepted by a DTA. In this section, we will deal with finite acceptance conditions, i.e., a DTA accepts the timed path if one of its final locations is reached. The results, however, also carry over to Muller acceptance conditions.

Deterministic timed automata. A DTA is a finite-state automaton equipped with a finite set of real-valued variables, called *clocks*. Clocks increase implicitly, all at the same pace, they can be inspected (in guards) and can be reset to the value zero. Let $\mathcal{X}$ be a finite set of clocks ranged over by x and y. A *clock constraint* g over set $\mathcal{X}$ is either of the form $x \bowtie c$ with $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, >, \geq\}$, or of the form $x - y \bowtie c$, or a conjunction of clock constraints. Let $\mathcal{CC}(\mathcal{X})$ denote the set of clock constraints over $\mathcal{X}$.

Definition 5.14 (DTA) *A* deterministic timed automaton *(DTA) is a tuple* $\mathcal{A} = (\Sigma, \mathcal{X}, Q, q_0, F, \rightarrow)$ *where:*

- Σ *is a finite alphabet,*
- $\mathcal{X}$ *is a finite set of clocks,*
- Q *is a nonempty finite set of locations with the initial location* $q_0 \in Q$*,*
- $F \subseteq Q$ *is a set of accepting (or final) locations,*
- $\rightarrow \subseteq Q \times \Sigma \times \mathcal{CC}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ *is the edge relation satisfying:*

$$(q \xrightarrow{a,g,X} q' \text{ and } q \xrightarrow{a,g',X'} q'' \text{ with } g \neq g') \quad \text{implies} \quad g \cap g' = \varnothing.$$

Intuitively, the edge $q \xrightarrow{a,g,X} q'$ asserts that the DTA $\mathcal{A}$ can move from location q to q' when the input symbol is a and the guard g holds, while the clocks in X should be reset when entering q' (all other clocks keep their value). DTA are deterministic as they have a single initial location, and outgoing edges of a location labeled with the same input symbol are required to have disjoint guards. In this way, the next location is uniquely determined for a given location and a given set of clock values. In case no guard is satisfied in a location for a given clock valuation, time can progress. If the advance of time will never reach a

situation in which a guard holds, the DTA will stay in that location *ad infinitum*. Note that DTA do not have location invariants.

The semantics of a DTA is given by an infinite-state transition system. We do not provide the full semantics, cf. [1], but we define the notion of paths, i.e., runs or executions of a DTA. This is done using some auxiliary notions. A *clock valuation* η for a set $\mathcal{X}$ of clocks is a function $\eta : \mathcal{X} \to \mathbb{R}_{\geq 0}$, assigning to each clock $x \in \mathcal{X}$ its current value $\eta(x)$. The clock valuation η over $\mathcal{X}$ satisfies the clock constraint g, denoted $\eta \models g$, iff the values of the clocks under η fulfill g. For instance, $\eta \models x - y > c$ iff $\eta(x) - \eta(y) > c$. Other cases are defined analogously. For $d \in \mathbb{R}_{\geq 0}$, $\eta{+}d$ denotes the clock valuation where all clocks of η are increased by d. That is, $(\eta{+}d)(x) = \eta(x){+}d$ for all clocks $x \in \mathcal{X}$. Clock *reset* for a subset $X \subseteq \mathcal{X}$, denoted by $\eta[X := 0]$, is the valuation η' defined by: $\forall x \in X.\eta'(x) := 0$ and $\forall x \notin X.\eta'(x) := \eta(x)$. We denote the valuation that assigns 0 to all the clocks by $\vec{0}$. An (infinite) path of DTA $\mathcal{A}$ has the form $\rho = q_0 \xrightarrow{a_0,t_0} q_1 \xrightarrow{a_1,t_1} \ldots$ such that $\eta_0 = \vec{0}$, and for all $j \geqslant 0$, it holds $t_j > 0$, $\eta_j{+}t_j \models g_j$, $\eta_{j+1} = (\eta_j{+}t_j)[X_j := 0]$, where η_j is the clock evaluation on entering q_j. Here, g_j is the guard of the j-th edge taken in the DTA and X_j the set of clock to be reset on that edge. A path ρ is accepted by $\mathcal{A}$ if $q_i \in F$ for some $i \geqslant 0$. Since the DTA is deterministic, the successor location is uniquely determined; for convenience we write $q' = \mathit{succ}(q, a, g)$. A path in a CTMC $\mathcal{C}$ can be "matched" by a path through DTA $\mathcal{A}$ by regarding sets of atomic propositions in $\mathcal{C}$ as input symbols of $\mathcal{A}$. Such path is accepted, if at some point an accepting location in the DTA is reached:

Definition 5.15 (CTMC paths accepted by a DTA) *Let CTMC* $\mathcal{C} = (S, R, AP, L, s_0)$ *and DTA* $\mathcal{A} = (2^{\mathrm{AP}}, \mathcal{X}, Q, q_0, F, \to)$. *The CTMC path* $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots$ *is accepted by* $\mathcal{A}$ *if there exists a corresponding DTA path*

$$q_0 \xrightarrow{L(s_0),t_0} \underbrace{\mathit{succ}(q_0, L(s_0), g_0)}_{=q_1} \xrightarrow{L(s_1),t_1} \underbrace{\mathit{succ}(q_1, L(s_1), g_1)}_{=q_2} \cdots$$

such that $q_j \in F$ *for some* $j \geqslant 0$. *Here,* $\eta_0 = \vec{0}$, g_i *is the (unique) guard in* q_i *such that* $\eta_i{+}t_i \models g_i$ *and* $\eta_{i+1} = (\eta_i{+}t_i)[X_i := 0]$, *and* η_i *is the clock evaluation on entering* q_i, *for* $i \geqslant 0$. *Let* $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \{\pi \in \mathit{Paths}^{\mathcal{C}} \mid \pi \text{ is accepted by DTA } \mathcal{A}\}$.

Theorem 5.16 ([31, 56]) *For any CTMC* $\mathcal{C}$ *and DTA* $\mathcal{A}$, *the set* $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ *is measurable.*

The main result of this theorem is that $Paths^{\mathcal{C}}(\mathcal{A})$ can be rewritten as the combination of cylinder sets of the form $Cyl = (s_0, I_0, \ldots, I_{n-1}, s_n)$ (Cyl for short) which are all accepted by DTA $\mathcal{A}$. A cylinder set (Cyl) is accepted by DTA $\mathcal{A}$ if all its paths are accepted by $\mathcal{A}$. That is

$$Paths^{\mathcal{C}}(\mathcal{A}) = \bigcup_{n\in\mathbb{N}} \bigcup_{\pi\in Paths^{\mathcal{C}}_n(\mathcal{A})} Cyl_\pi, \tag{5.1}$$

where $Paths^{\mathcal{C}}_n(\mathcal{A})$ is the set of accepting paths by $\mathcal{A}$ of length n and Cyl_π is the cylinder set that contains π.

Definition 5.17 (WL related cylinder sets) *Two cylinder sets $Cyl = (s_0, I_0, \ldots, I_{n-1}, s_n)$ and $Cyl' = (s'_0, I_0, \ldots, I_{n-1}, s'_n)$ are said to be WL related, denoted $Cyl \cong Cyl'$, if they are statewise WL related: $Cyl \cong Cyl'$ iff $s_i \cong s'_i$ for all $0 \leqslant i \leqslant n$.*

Definition 5.18 (WL-closed set) *The set Π of cylinder sets is WL-closed if $\forall Cyl \in \Pi$, and Cyl' with $Cyl' \cong Cyl$ implies $Cyl' \in \Pi$.*

A finite path π in the CTMC $\mathcal{C}$ is compatible with Π if the cylinder set for this path $Cyl_\pi \in \Pi$. Since the cylinder sets contained in Π are disjoint, we have $\Pr_s(\Pi) = \Pr_s(\bigcup_{Cyl\in\Pi} Cyl) = \sum_{Cyl\in\Pi} \Pr_s(Cyl)$, where $\Pr_s(\Pi)$ is the probability of all the paths starting in s which are compatible with Π. For paths compatible with Π but not starting from s, the probability equals 0. We denote WL-closed set of cylinder sets of length n by Π_n. If $n = 0$, Π_n is the set of states and $\Pr_s(\Pi_n) = \alpha(s)$ if $s \in \Pi_n$, 0 otherwise, where $\alpha(s)$ is the probability of s being the initial state of CTMC $\mathcal{C}$.

Example 5.19 *Consider the CTMCs given in Fig. 5.6, here we have the CTMC $\mathcal{C}$ (left) and its quotient system $\mathcal{C}/_{\mathcal{R}}$ (right). In this case if $\Pi = \{Cyl(s_0, I_0, s_1, I_1, s_3), Cyl(s'_0, I_0, s'_1, I_1, s'_2)\}$ is a WL closed set of cylinder sets in $\mathcal{C}$, and $\mathcal{C}/_{\mathcal{R}}$ that are accepted by DTA $\mathcal{A}$, then:*

$$\begin{aligned}\Pr_{s_0}(\Pi) &= \Pr_{s_0}(Cyl(s_0, I_0, s_1, I_1, s_3)) + \Pr_{s_0}(Cyl(s'_0, I_0, s'_1, I_1, s'_2)) \\ &= 1/2 \cdot (e^{-E(s_0)\cdot \inf I_0} - e^{-E(s_0)\cdot \sup I_0}) \cdot (e^{-E(s_1)\cdot \inf I_1} - e^{-E(s_1)\cdot \sup I_1}) + 0.\end{aligned}$$

The second term is 0 as the cylinder set does not start from s_0. Similarly,

$$\begin{aligned}\Pr_{s'_0}(\Pi) &= \Pr_{s'_0}(Cyl(s_0, I_0, s_1, I_1, s_3)) + \Pr_{s'_0}(Cyl(s'_0, I_0, s'_1, I_1, s'_2)) \\ &= 0 + (e^{-E(s'_0)\cdot \inf I_0} - e^{-E(s'_0)\cdot \sup I_0}) \cdot 1/2 \cdot (e^{-E(s'_1)\cdot \inf I_1} - e^{-E(s'_1)\cdot \sup I_1}).\end{aligned}$$

Definition 5.20 *For CTMC $\mathcal{C}$ and DTA $\mathcal{A}$, let* $\Pr(\mathcal{C} \models \mathcal{A}) = \Pr\left(Paths^{\mathcal{C}}(\mathcal{A})\right)$.

Stated in words, $\Pr(\mathcal{C} \models \mathcal{A})$ denotes the probability of all the paths in CTMC $\mathcal{C}$ that are accepted by DTA $\mathcal{A}$. Note that we slightly abuse notation, since Pr on the right-hand side is the probability measure on the Borel space of infinite paths in the CTMC. This brings us to one of the main results of this chapter:

Theorem 5.21 (Preservation of DTA specifications) *For any CTMC $\mathcal{C}$, a WL $\mathcal{R}$ on $\mathcal{C}$ and DTA $\mathcal{A}$:*

$$\Pr(\mathcal{C} \models \mathcal{A}) \;=\; \Pr(\mathcal{C}/_{\mathcal{R}} \models \mathcal{A}).$$

The detailed proof is in the appendix and consists of two main steps:

1. We prove that for any cylinder set *Cyl* in the quotient CTMC $\mathcal{C}/_{\mathcal{R}}$ which is accepted by the DTA $\mathcal{A}$, there is a corresponding *set* of cylinder sets in the CTMC $\mathcal{C}$ that are acccepted by the DTA $\mathcal{A}$ and that jointly have the same probability as *Cyl*, cf. Lemma 5.22 below.

2. We show that the sum of probabilities of all the cylinder sets in $\mathcal{C}/_{\mathcal{R}}$ that are accepted by DTA $\mathcal{A}$ equals the sum of probabilities of all the corresponding sets of cylinder sets in $\mathcal{C}$.

Lemma 5.22 *Let $\mathcal{C} = (S, R, AP, L, s_0)$ be a CTMC and $\mathcal{R}$ be a WL on $\mathcal{C}$. If Π is a WL-closed set of cylinder sets which are accepted by DTA $\mathcal{A}$, then for any $D \in S/_{\mathcal{R}}$ and $s_0' \in Pred(D)$:*

$$\sum_{s_1 \in D} P(s_0', s_1, D) \cdot \Pr_{s_1}(\Pi) = \Pr_{D}(\Pi).$$

From Lemma 5.22 we conclude

$$\sum_{D \in S/_{\mathcal{R}}} \sum_{s_1 \in D} P(s_0', s_1, D) \cdot \Pr_{s_1}(\Pi) = \sum_{D \in S/_{\mathcal{R}}} \Pr_{D}(\Pi). \tag{5.2}$$

Corollary 5.23 *WL preserves transient state probabilities.*

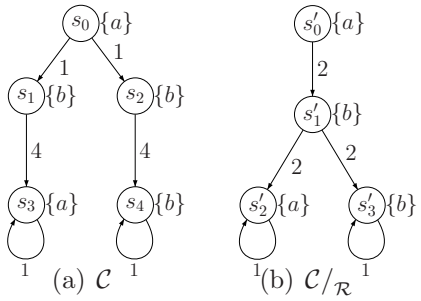

Figure 5.6: WL related cylinder sets.

5.1.4 Preservation of MTL Specifications

In this section we show that the quotient CTMC obtained under WL can be used for verifying Metric Temporal Logic (MTL) formulae [24, 95, 125]. It is interesting to note that the expressive power of MTL is different from that of DTA. Temporal properties like $(\Diamond\Box a)$ cannot be expressed using deterministic timed automata, since nondeterminism is needed to compensate for the non causality [109]. On the other hand, DTA expressible languages that involve counting [9], e.g., a should only occur at even positions, cannot be expressed using MTL. We now recall the syntax and semantics of Metric Temporal Logic [24, 125].

Definition 5.24 (Syntax of MTL) *Let AP be a set of atomic propositions, then the formulas of MTL are built from AP using Boolean connectives, and time-constrained versions of the* until *operator* U *as follows:*

$$\varphi \ ::= \ tt \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathrm{U}^{I}\, \varphi$$

where $I \subseteq \mathbb{R}_{\geq 0}$ is a nonempty interval with rational bounds, and $a \in AP$.

Whereas, typically, the semantics of MTL is defined over timed paths of timed automata, we take a similar approach by interpreting MTL formulas over CTMC paths.

Definition 5.25 (Semantics of MTL formulas) *The meaning of MTL formulas is defined by means of a satisfaction relation, denoted by $\models$, between a CTMC $\mathcal{C}$, one of its paths π, MTL formula φ, and time $t \in \mathbb{R}_{\geq 0}$. Let $\pi = s_0 \xrightarrow{t_0} s_1 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \cdots$ be a finite or infinite path of $\mathcal{C}$, then $(\pi, t) \models \varphi$*

is defined inductively by:

$$
\begin{array}{lcll}
(\pi,t) & \models & tt & \\
(\pi,t) & \models & a & \textit{iff} \quad a \in L(\pi@t) \\
(\pi,t) & \models & \neg\varphi & \textit{iff} \quad \textit{not } (\pi,t) \models \varphi \\
(\pi,t) & \models & \varphi_1 \wedge \varphi_2 & \textit{iff} \quad (\pi,t) \models \varphi_1 \textit{ and } (\pi,t) \models \varphi_2 \\
(\pi,t) & \models & \varphi_1 \,\mathrm{U}^I\, \varphi_2 & \textit{iff} \quad \exists t' \in t{+}I.\ ((\pi,t') \models \varphi_2 \wedge \forall t \leq t'' < t'.\, (\pi,t'') \models \varphi_1)\,.
\end{array}
$$

The semantics for the propositional fragment is straightforward. Recall that $\pi@t$ denotes the state occupied along path π at time t. Path π at time t satisfies $\varphi_1 \,\mathrm{U}^I\, \varphi_2$ whenever for some time point t' in the interval $I{+}t$, defined as $[a,b]{+}t = [a{+}t, b{+}t]$ (and similarly for open intervals), φ_2 holds, and at all time points between t and t', path π satisfies φ_1. Let $\pi \models \varphi$ if and only if $(\pi,0) \models \varphi$. The standard temporal operators like $\Diamond$ ("eventually") and its timed variant $\Diamond^I$ are derived in the following way: $\Diamond^I\varphi = tt\,\mathrm{U}^I\,\varphi$ and$\Diamond\varphi = tt\,\mathrm{U}\,\varphi$. Similarly, $\Box$ ("globally") and its timed variant are derived as follows:

$$\Box^I\varphi = \neg(\Diamond^I\neg\varphi) \text{ and } \Box\varphi = \neg(\Diamond\neg\varphi).$$

Example 5.26 *Using MTL, various interesting properties can be specified such as:*

- $\Box(\mathit{down} \rightarrow \Diamond^{[0,5]}\mathit{up})$*, which asserts that whenever the system is down, it should be up again within 5 time units.*
- $\Box(\mathit{down} \rightarrow \mathit{alarm}\,\mathrm{U}^{[0,10]}\,\mathit{up})$*, which states that whenever the system is down, an alarm should ring until it is up again within 10 time units.*

More complex properties can be specified by nesting of until path formulas.

Theorem 5.27 ([11]) *The probability measure of the set of converging paths is zero.*

As a next result, we address the measurability of a set of CTMC paths satisfying an MTL formula φ.

Theorem 5.28 *For each MTL formula φ and state s of CTMC $\mathcal{C}$, the set $\{\pi \in Paths(s) \mid \pi \models \varphi\}$ is measurable.*

Definition 5.29 (Probability of MTL formulas) *The probability that state s satisfies MTL formula φ refers to the probability for the sets of paths for which that formula holds as follows:*

$$\Pr(s \models \varphi) \;=\; \Pr_{s}(\pi \in \mathit{Paths}(s) \mid \pi \models \varphi).$$

Since $\mathcal{C}$ has a single initial state, i.e., s_0, the probability of all the paths in $\mathcal{C}$ that satisfy MTL formula φ is given by $\Pr(\mathcal{C} \models \varphi) = \Pr(s_0 \models \varphi)$.

Theorem 5.30 *Let $\mathcal{C}$ be a CTMC and $\mathcal{R}$ be a WL on $\mathcal{C}$. Then for any MTL formula φ:*

$$\Pr(\mathcal{C} \models \varphi) = \Pr(\mathcal{C}/_{\mathcal{R}} \models \varphi).$$

5.2 Case Studies

In [159], a first attempt has been made towards developing and implementing a polynomial-time algorithm for WL quotienting. The worst-case time complexity of this algorithm is $\mathcal{O}(n^5)$. It performs repeated minimization on the input CTMC until no further minimization is possible. This algorithm can be easily adapted to compute KME on KSs and WPE on DTMCs. Although the algorithm is slow and there is still room for improvement, initial experiments show that it can substantially reduce the size of the CTMC models for incremental service case studies. Note that for incremental service systems bisimulation usually fails to provide any state space reduction [17]. In this section we discuss the experimental results obtained for two case studies namely, a restaurant system and a job-server system. Both the case studies have been modeled and analyzed using the PRISM model checker. Since PRISM does not support bisimulation minimization, a separate procedure was written for computing the quotient under bisimulation. All the tests were executed on a Windows 7 machine with a Core2Duo E6550 CPU (2.33 GHz) and 2 GB RAM. The execution times are averaged over five test runs.

5.2.1 Restaurant System

The idea behind this case study is very similar to [17]. Consider a restaurant where a person can have single or multiple courses. For each course he or she has to go through three phases namely, ordering, waiting and eating. The guest is supposed to wait for the waiter until his or her order is taken. Once the

n	input CTMC	bisim.	red. bisim.	WL	red. WL	execution time
50	3927	3829	97.5%	154	3.9%	1:15.9
75	8702	8554	98.3%	229	2.6%	8:08.8
100	15352	15154	98.7%	304	2.0%	31:41.8

Table 5.1: State space reduction for the restaurant system

l	q	input CTMC	WL quot.	red. WL	execution time
3	3	1276	847	66.4%	6.0
	4	4835	3224	66.7%	50.5
	5	17466	11647	66.7%	25:29.9
4	3	4076	2164	53.1%	35.6
	4	20455	10879	53.2%	15:44.8

Table 5.2: State space reduction for the job-server system

food is ready, the guest can start eating his or her food. After finishing the last course, the guest should again wait for the waiter to bring the bill and then pays the money. It is assumed that infinitely many guests can eat in the restaurant one after another. The main principle is captured in Fig. 5.7 where rates are associated with every phase of the system. More specifically, let $\lambda_o, \lambda_s, \lambda_f$ and λ_p denote the rates for the time required to order food, serve food, finish eating and pay the bill, respectively. The rates for each these phases are assumed to remain same throughout the whole CTMC. The results for different values of n are summarised in Table 5.1. Here n denotes the maximum number of courses that can be ordered.

5.2.2 Job-Server System

Consider a server system that can process jobs which are added to a queue. Every job needs several execution steps before it is completed. It is assumed that jobs can be instantly added to the queue and the enqueueing of the job is synchronized with the working of the server. Intuitively this means that each time an execution step is finished by the server, either a new job is added to the queue with parameter n or no job is added. Here n denotes the number of execution steps required for the new job that has been enqueued. The results for different values of l and q are summarised in Table 5.2. Here l is the maximum number of execution steps required for processing a job and q is the length of the queue. For this case study the original system and bisimulation quotient had the

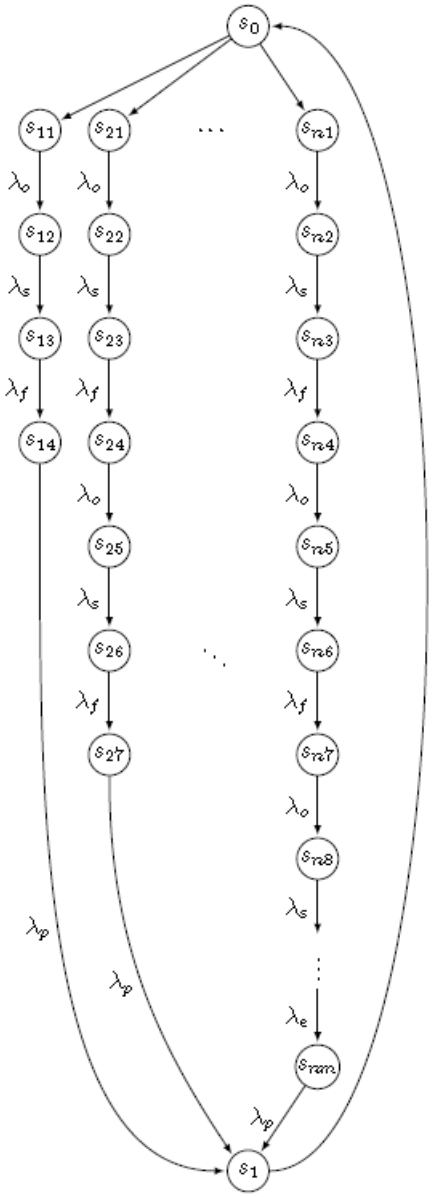

Figure 5.7: Restaurant system

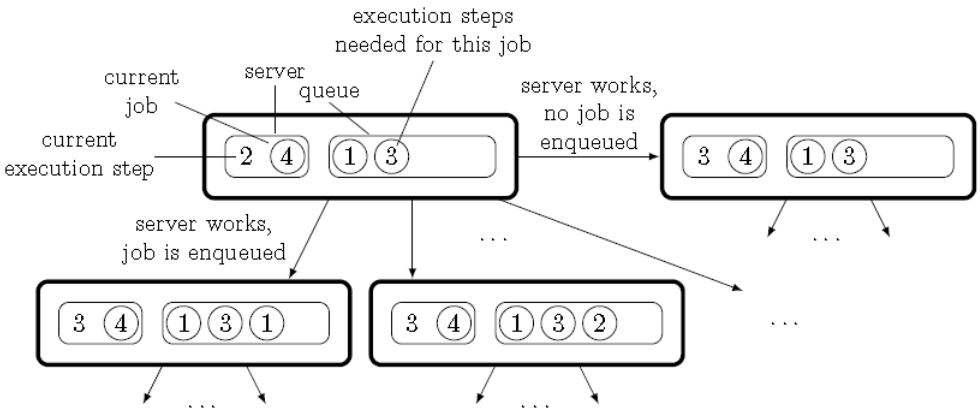

Figure 5.8: Job-server system

same number of states and transitions for all values of l, q.

5.3 Related Work

Various branching-time relations on CTMCs have been defined such as weak and strong variants of bisimulation equivalence and simulation pre-orders [10, 11, 29, 93]. Strong bisimulation coincides with ordinary lumping equivalence [29]. Their compatibility to (fragments of) stochastic variants of CTL has been thoroughly investigated, cf. [10]. More specifically, strong bisimulation for CTMCs coincides with equivalence for CSL. Similarly, weak bisimulation for CTMCs coincides with equivalence for CSL without next operator. Note that strong bisimulation coincides with strong simulation equivalence for CTMCs. In the weak setting, weak bisimulation coincides with weak simulation equivalence for CTMCs. In addition, it has been proved that strong simulation and weak simulation for CTMCs, preserve a safe fragment of CSL and CSL without next operator, respectively [10]. These relations allow for a state-space reduction prior to model checking; in particular, bisimulation minimisation yields considerable reductions and time savings [91] thanks to an efficient minimisation algorithm [48, 154].

This chapter focuses on a notion of lumpability that allows for a more aggressive state-space aggregation than ordinary lumpability. It originates by Bernardo [16] who considered Markovian testing equivalence over sequential Markovian process calculus (SMPC), and coined the term T-lumpability for the induced state-level aggregation where T stands for testing. His testing equiva-

lence coincides with ready trace equivalence on CTMCs [157], it is a congruence w.r.t. parallel composition, and preserves transient as well as steady-state probabilities [16]. A logical characterisation via a variant of Hennessy-Milner logic has been given in [18, 19] establishing the preservation of expected delays. Bernardo defines T-lumpability using four process-algebraic axioms, and alternatively, calls two states T-lumpable if their expected delays w.r.t. any testing process coincide.

5.4 Conclusions

This chapter considered weighted lumpability (WL), a structural notion that coincides with Bernardo's T-lumpability [16] which was defined in a process-algebraic setting. Whereas Bernardo defines T-lumpability in an axiomatic manner, our starting point is a structural definition using first CTMC principles. The main

Equivalence	(stutter) ω-regular (ω)	DTA/MTL	Compositional (C)
KME	ω		C
WKME	stutter-insensitive ω		
WPE	ω		C
WL		DTA+MTL	

Table 5.3: An overview of various equivalence relations

contribution of this chapter is the preservation of DTA and MTL specifications under WL quotienting. We note that this implies the preservation of transient probabilities as well as timed reachability probabilities. We show that MTL formulas (interpreted on CTMCs) are measurable. We have also presented some case studies showing that WL can substantially reduce the size of the CTMC state space. The results presented in this chapter can also be extended to CTMCs with rewards. A summary of all the equivalence relations discussed in Chapter 3, Chapter 4 and Chapter 5 is given in Table 5.3.

Chapter 6

Layered Reduction for Modal Specification Theories

Modal transition system (MTS) specifications support compositionality together with a compositional step-wise refinement methodology, and thus are useful for component-oriented design and analysis of distributed systems. In this setting, a high-level model of the system which abstracts from the implementation details is constructed and used for the verification of interesting properties. A correct implementation can be obtained by applying a series of successive refinement steps. Model construction involves composing several components in parallel, where each component usually has multiple sub-components that are executed in a sequential manner. Components cooperate through their synchronization over common actions and through their respective action dependencies. Action dependencies between sub-components can be either explicitly stated or derived from the operations performed on shared data variables that are updated during an action execution (in case of MTS with data [13]). Some example systems that have this structure are distributed algorithms such as the distributed minimum weight spanning tree algorithm [58], the two phase commit protocol [20], Fischer's real-time mutual exclusion protocol, and the randomized mutual exclusion algorithm by Kushilevitz and Rabin [96]. Composing several components using parallel composition naturally leads to the problem of *state-space explosion* [9], where the number of states grows exponentially in the number of parallel components.

This chapter proposes a state-space reduction technique based on the notion of layering [82, 145] for a network of acyclic MTSs. The main principle is illustrated in Fig. 6.1. Here two acyclic MTS components $\mathcal{M}$ and $\mathcal{N}$ are composed in parallel

(left), where each component consists of n sub-components which are executed in a sequential manner (denoted by ;). The system obtained after performing layered transformation is shown in Fig. 6.1 (right). All the sub-components of $\mathcal{M}$ and $\mathcal{N}$ are assumed to be acyclic, and can be repeated by allowing top-level recursion in $\mathcal{M}$ and $\mathcal{N}$ (as indicated by *). In other words, every component ($\mathcal{M}$ and $\mathcal{N}$) can have multiple rounds of execution, where a new round is started only when the last sub-component of the previous round, i.e., $\mathcal{M}_n$) has been completed. This is important as deadlocks are usually considered to be undesirable for distributed algorithms.

Roughly speaking, layering exploits the independence between sub-components to transform the system under consideration from a distributed representation to a layered representation. These transformations are syntactic: the idea is to apply a series of transformations to model descriptions, yielding a layered representation (cf. Fig. 6.1 (right)). For the intermediate transformations we use a layered composition operator denoted by "$\bullet$". Informally, $\mathcal{M}_1 \bullet \mathcal{M}_2$ allows synchronization on common actions and interleaving on disjoint actions, except when some action a of $\mathcal{M}_2$ depends on one or more actions of $\mathcal{M}_1$; in this case, a can be executed only after all the actions of $\mathcal{M}_1$ on which it depends have been executed. This new composition operator allows formulating *Communication Closed Layer* (CCL) laws [51, 82], which are required to carry out the structural transformations and establish an equivalence between the two systems. Since the sub-components within a component are executed sequentially, a partial order relation is proposed to relate the $\bullet$ and ; (sequential) operators.

The reduced system obtained as a result of applying layered transformation can be used for analysis, provided it preserves a rich class of properties of interest. Reachability is one of the most important properties in the area of model checking [9]. Therefore, we focus on proving that the reduced system preserves existential ($\exists$) and universal ($\forall$) reachability properties for a set of final states. As a result, existential and universal reachability properties can be checked on the layered, typically smaller, model.

Contributions. The main contributions of this chapter are as follows:

- We define the layered ($\bullet$) and sequential (;) composition operator, and formulate communication closed layer (CCL) laws for acyclic MTSs.
- We define partial order (po) equivalence between acyclic MTSs, show that $\bullet$ is po-equivalent to ;, and prove that it preserves existential ($\exists$) and universal ($\forall$) reachability properties.

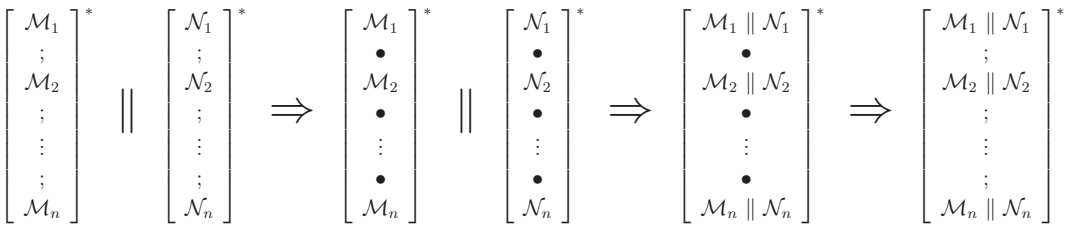

Figure 6.1: Layered reduction

- Finally, we show how state space reduction can be achieved by replacing $\bullet$ with ; within the framework of CCL laws.

Organisation of this chapter. Section 6.1 presents the satisfaction and refinement relations for MTSs. Section 6.2 discusses the composition operators for MTSs, and introduces CCL laws. Section 6.3 defines po-equivalence between MTSs, and proves that po-equivalence between $\bullet$ and ; preserves existential ($\exists$) and universal ($\forall$) reachability properties. Section 6.4 discusses the possible extensions of our results. Section 6.5 discusses related work. Finally, Section 6.6 concludes the chapter.

6.1 Satisfaction and Refinement

This section presents the notions of *satisfaction* and *refinement* originally introduced in [106]. A satisfaction relation allows to relate an LTS (implementation) with an MTS (specification). A refinement relation is used to compare MTSs w.r.t. their sets of implementations. We also define the notions of realisation, existential ($\exists$) and universal ($\forall$) reachability properties of reaching the set of final states computed over the implementations of an MTS.

Definition 6.1 *(Satisfaction) Let $\mathcal{T} = (S, Act, s_0, S_f, V)$ be an LTS and $\mathcal{M} = (S', Act, s'_0, S'_f, V')$ be an MTS. $R \subseteq S \times S'$ is a* satisfaction relation *iff, for any $(s, s') \in R$, the following conditions hold:*

- $\forall a \in Act, \forall u' \in S' : V'(s', a, u') = \top \Rightarrow (\exists u \in S : V(s, a, u) = \top \wedge uRu')$,
- $\forall a \in Act, \forall u \in S : V(s, a, u) = \top \Rightarrow (\exists u' \in S' : V'(s', a, u') \neq \bot \wedge uRu')$,
- $s \in S_f \Leftrightarrow s' \in S'_f$.

We say that $\mathcal{T}$ satisfies $\mathcal{M}$, denoted $\mathcal{T} \models \mathcal{M}$, iff there exists a satisfaction relation relating s_0 and s'_0. If $\mathcal{T} \models \mathcal{M}$, $\mathcal{T}$ is called an implementation *of $\mathcal{M}$. Let $[\![\mathcal{M}]\!] = \{\mathcal{T} \mid \mathcal{T} \models \mathcal{M}\}$, the set of implementations of MTS $\mathcal{M}$.*

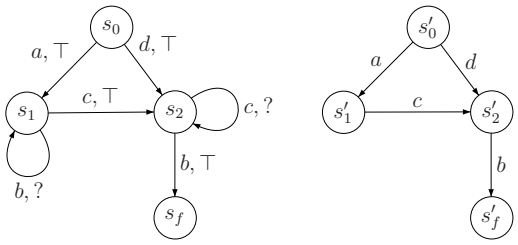

Figure 6.2: An MTS $\mathcal{M}$ (left) and an LTS $\mathcal{T}$ (right) such that $\mathcal{T} \models \mathcal{M}$

Intuitively, a state s in LTS $\mathcal{T}$ satisfies state s' in MTS $\mathcal{M}$ iff any must transition of s' is matched by a transition of s, and s does not contain any transitions without a corresponding transition (may or must) in s'. In addition, final states of $\mathcal{T}$ can only be related to final states in $\mathcal{M}$ and vice versa.

Definition 6.2 *Let $\mathcal{T} \in [\![\mathcal{M}]\!]$, and $\pi = s'_0 a'_1 s'_1 a'_2 s'_2 \dots s'_n$ be a finite path of $\mathcal{T}$, i.e., $\pi \in Paths_{fin}(\mathcal{T})$. π is said to be a* realisation *of $\rho = s_0 a_1 s_1 a_2 s_2 \dots s_n$ where $\rho \in Exec_{fin}(\mathcal{M})$, denoted $\pi \models \rho$, if $\forall i < n : a_{i+1} = a'_{i+1}$.*

Example 6.3 *The LTS $\mathcal{T}$ in Fig. 6.2 (right) is an implementation of the MTS $\mathcal{M}$ in Fig. 6.2 (left). It is easy to check that there exists a satisfaction relation relating the initial states of $\mathcal{T}$ and $\mathcal{M}$. Note that in this example, for every implementation of $\mathcal{M}$, $S_f \neq \varnothing$ (since there exists a finite execution from s_0 to s_f with only must transitions).*

Example 6.4 *Finite path $\pi = s'_0 a s'_1 c s'_2 b s'_f$ of LTS $\mathcal{T}$ (Fig. 6.2 (right)) is a realisation of finite execution $\rho = s_0 a s_1 c s_2 b s_f$ of $\mathcal{M}$ (Fig. 6.2 (left)).*

Note that for a deterministic MTS $\mathcal{M}$ with $\mathcal{T} \models \mathcal{M}$, if a path $\pi \in Paths_{fin}(\mathcal{T})$ is a realisation of $\rho \in Exec_{fin}(\mathcal{M})$, then π cannot be a realisation of another execution of $\mathcal{M}$.

Definition 6.5 *(Refinement) Let $\mathcal{M} = (S, Act, s_0, S_f, V)$ and $\mathcal{M}' = (S', Act, s'_0, S'_f, V')$ be MTSs. $R \subseteq S \times S'$ is a strong* refinement relation *iff, for all $(s, s') \in R$, the following conditions hold:*

- $\forall a \in Act, \forall u' \in S' : V'(s', a, u') = \top \Rightarrow (\exists u \in S : V(s, a, u) = \top \wedge uRu')$,
- $\forall a \in Act, \forall u \in S : V(s, a, u) \neq \bot \Rightarrow (\exists u' \in S' : V'(s', a, u') \neq \bot \wedge uRu')$,

- $s \in S_f \Leftrightarrow s' \in S'_f$.

$\mathcal{M}$ strongly refines $\mathcal{M}'$, denoted $\mathcal{M} \preceq_S \mathcal{M}'$, iff there exists a strong refinement relation relating s_0 and s'_0.

Intuitively, a state s strongly refines state s' iff any must transition of s' is matched by a must transition of s, s does not contain any transitions (may or must) without a corresponding transition (may or must) in s'. In addition, final states of $\mathcal{M}$ can only be related to final states in $\mathcal{M}'$ and vice versa.

Remark 6.6 *A satisfaction relation can be seen as a special case of refinement relation. In simple words, if $\mathcal{T} \models \mathcal{M}$, then $\mathcal{T} \preceq_S \mathcal{M}$ (since every LTS is an MTS and all the three conditions of refinement are satisfied).*

Definition 6.7 *(Refinement equivalence) MTSs $\mathcal{M}$ and $\mathcal{M}'$ are* refinement equivalent*, denoted $\mathcal{M} \equiv \mathcal{M}'$, iff $\mathcal{M} \preceq_S \mathcal{M}'$ and $\mathcal{M}' \preceq_S \mathcal{M}$.*

Since strong refinement implies inclusion of sets of implementations, if $\mathcal{M} \equiv \mathcal{M}'$ then they have the same set of implementations, i.e., $[\![\mathcal{M}]\!] = [\![\mathcal{M}']\!]$.
Assumptions. For the rest of the chapter, we assume:

- Every MTS is acyclic.
- Every MTS has a single final state, i.e., $|S_f| = 1$, and all its states (except the final state) have at least one outgoing transition.
- Dependencies between actions of different components are known in advance.

The acyclicity assumption is required to establish an equivalence between the layered operator ($\bullet$) and sequential operator (;). This restriction is less limiting than may appear at first sight, since in many distributed systems the sub-components/phases are acyclic [82]. As mentioned before, we do allow the algorithm to be executed multiple times, i.e., top-level recursion is allowed. The second assumption ensures that deadlock states (which are usually undesirable) are absent. Moreover, this assumption ensures that the control is transferred from one component to another via a single state. This usually happens in case of distributed algorithms, e.g., randomized mutual exclusion algorithm [149], Fischer's real-time mutual exclusion protocol [85, 124], and two phase commit protocol [82]. The dependency relation between actions of sub-components/phases can be

either explicitly stated or derived from the operations performed on data variables that are updated during an action execution [82]. For instance, a read access to a shared variable in a component depends on a write access of that variable in another component. More formally, two actions a and b are dependent in an MTS $\mathcal{M}$ extended with data variables, denoted $a \dagger b$, if any one of the following holds:

$$Write(b) \cap Read(a) \neq \varnothing,$$
$$Write(a) \cap Read(b) \neq \varnothing,$$
$$Write(a) \cap Write(b) \neq \varnothing.$$

Here, $Write(a)$ denotes the set of data variables written by the action a. Similarly, $Read(a)$ denotes the set of data variables read by the action a. Since we consider MTSs without data variables, we assume that the dependencies between actions are explicitly stated by the modeler. In this chapter we focus on reachability properties, i.e., is it possible to reach the set of final states from the initial state in an LTS $\mathcal{T}$. More formally it is defined as follows:

Definition 6.8 (LTS reachability) *Let $\mathcal{T} = (S, Act, s_0, S_f, V)$ be an LTS. Then $\mathcal{T}$ reaches S_f, denoted $\mathcal{T} \models \Diamond S_f$, iff $\forall \pi \in Paths_{fin}(\mathcal{T}) \exists \pi' \in Paths_{fin}^{S_f}(\mathcal{T})$: π is a prefix of π'.*

In simple words, all the finite paths starting from the initial state of $\mathcal{T}$ should be extendable such that the last state of the new path obtained belongs to S_f.

Example 6.9 *Consider the LTS $\mathcal{T}$ in Fig. 6.2 (right) where s_0 is the initial state, and s_f is the only final state. Here, $\mathcal{T} \models \Diamond S_f$ since every finite path of $\mathcal{T}$ can be extended such that it reaches s_f.*

Next, we define two reachability properties of reaching the set of final states computed over all the implementations of an MTS $\mathcal{M}$. The first property requires that for an MTS $\mathcal{M}$ there exists at least one implementation $\mathcal{T}$ such that $\mathcal{T} \models \Diamond S_f$. The second property requires that all the implementations of $\mathcal{M}$ should be able to reach the set of final states. Formally, these properties are defined as follows:

Definition 6.10 (Existential reachability) *Let $\mathcal{M} = (S, Act, s_0, S_f, V)$ be an MTS. Then $\mathcal{M}$* possibly reaches *S_f, denoted $\mathcal{M} \models^{\exists} \Diamond S_f$, iff $\exists \mathcal{T} \in [\![\mathcal{M}]\!] : \mathcal{T} \models \Diamond S_f$.*

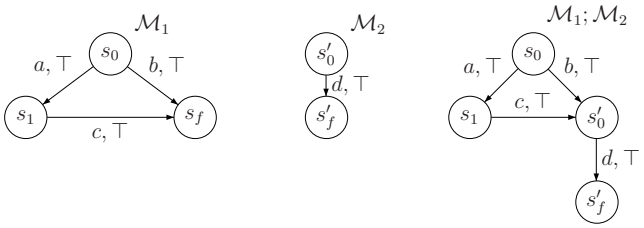

Figure 6.3: MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$ (left and middle) and their sequential composition (right)

Definition 6.11 (Universal reachability) *Let* $\mathcal{M} = (S, Act, s_0, S_f, V)$ *be an MTS. Then* $\mathcal{M}$ inevitably reaches S_f, *denoted* $\mathcal{M} \models^{\forall} \Diamond S_f$, *iff* $\forall \mathcal{T} \in [\![\mathcal{M}]\!] : \mathcal{T} \models \Diamond S_f$.

The problem of deciding $\mathcal{M} \models^{\exists} \Diamond S_f$ is PSPACE-complete [14]. The same applies to universal reachability.

6.2 Composition and CCL Laws

In this section we define composition operators for MTSs. We propose sequential, and layered composition operators, and recall parallel composition from [106]. The framework of CCL laws is also formulated, which is required for carrying out the layered transformations.

Definition 6.12 (Sequential composition) *Given MTSs* $\mathcal{M}_i = (S_i, Act_i, s_{0i}, \{s_{fi}\}, V_i)$, *where* $i \in \{1, 2\}$ *with* $S_1 \cap S_2 = \varnothing$. *The* sequential composition *of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$, *denoted* $\mathcal{M}_1; \mathcal{M}_2$, *is the MTS* $(S, Act_1 \cup Act_2, s_{01}, \{s_{f2}\}, V)$, *where* $S = S_1 \setminus \{s_{f1}\} \cup S_2$ *and* $V = V'_1 \cup V_2$. *Here* $V'_1 = V_1[s_{02} \leftarrow s_{f1}]$ *is defined by*

$$V'_1(s, a, s') = V_1(s, a, s') \text{ if } s' \neq s_{f1}, \text{ and}$$
$$V'_1(s, a, s_{02}) = V_1(s, a, s_{f1}) \text{ otherwise.}$$

Intuitively, sequential composition of two MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$ requires that the actions of $\mathcal{M}_2$ can only be executed once the final state of $\mathcal{M}_1$, i.e., s_f has been reached. Note that all the incoming transitions to state s_{f1} are redirected to s_{02}. Here, s_{01}, s_{f2} are the new initial and final states in the resulting MTS $\mathcal{M}_1; \mathcal{M}_2$, respectively.

Example 6.13 *The sequential composition of two MTSs* $\mathcal{M}_1, \mathcal{M}_2$ *(Fig. 6.3 (left)) is shown in Fig. 6.3 (right).*

Definition 6.14 (Parallel composition) *Given MTSs* $\mathcal{M}_i = (S_i, Act_i, s_{0i}, \{s_{fi}\}, V_i)$*, where* $i \in \{1,2\}$ *with* $S_1 \cap S_2 = \varnothing$*. The* parallel composition *of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*, denoted* $\mathcal{M}_1||\mathcal{M}_2$*, is the MTS* $(S_1 \times S_2, Act_1 \cup Act_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, V)$ *where for all* $(s, s') \in S_1 \times S_2$*,* V *is defined by:*

- *For all* $a \in Act_1 \cap Act_2$*, if there exists* $u \in S_1$ *and* $u' \in S_2$*, such that* $V_1(s,a,u) \neq \bot$ *and* $V_2(s',a,u') \neq \bot$*, define* $V((s,s'),a,(u,u')) = V_1(s,a,u) \sqcap V_2(s',a,u')$*. If either* $\forall u \in S_1$*, we have* $V_1(s,a,u) = \bot$*, or* $\forall u' \in S_2$*, we have* $V_2(s',a,u') = \bot$*, then* $\forall (u,u') \in S_1 \times S_2$*,* $V((s,s'),a,(u,u')) = \bot$*.*
- *For all* $(u,u') \in S_1 \times S_2$*,* $a \in Act_1 \setminus Act_2$*, define* $V((s,s'),a,(u,u')) = V_1(s,a,u)$ *if* $s' = u'$*,* $V((s,s'),a,(u,u')) = \bot$ *otherwise.*
- *For all* $(u,u') \in S_1 \times S_2$*,* $a \in Act_2 \setminus Act_1$*, define* $V((s,s'),a,(u,u')) = V_2(s',a,u')$ *if* $s = u$*,* $V((s,s'),a,(u,u')) = \bot$ *otherwise.*

Parallel composition forces synchronization on all common actions and interleaving on disjoint actions. The synchronization of two must transitions results in a must transition, and composing may-must, must-may and may-may transitions results in a may transition. Note that $||$ is commutative and associative.

Example 6.15 *The parallel composition of two MTSs* $\mathcal{M}_1, \mathcal{M}_2$ *(Fig. 6.3 (left and middle)) is shown in Fig. 6.4 (left).*

Next, we introduce the notion of action independence which is subsequently used to define layered composition. Let $a(s)$ denote the unique state that can be reached from state s in one step (may or must) by performing action $a \in act(s)$ in a deterministic MTS $\mathcal{M}$. The dependency between two actions a and b is denoted $a \dagger b$. Two additional requirements for the dependency relation $\dagger$ are that it is reflexive and symmetric. Two distinct actions that are not dependent are said to be independent, where independence is defined as follows:

Definition 6.16 *Let MTS* $\mathcal{M} = (S, Act, s_0, \{s_f\}, V)$*. Actions* $a, b \in Act$ *such that* $a \neq b$ *are* independent *in* $\mathcal{M}$*, denoted* $a \ddagger b$*, iff for all states* $s \in S$ *with* $a, b \in act(s)$ *we have:*

$$b \in act(a(s)), a \in act(b(s)), \text{ and } a(b(s)) = b(a(s)).$$

The first two conditions assert that a and b should not disable each other. The last condition asserts that the same state should be reached from s by either performing a followed by b, or by performing b followed by a. Let $act(s) \ddagger a$ denote $\forall b \in act(s) : b \ddagger a$. This notion of action independence originates from partial-order reduction [62, 128].

Definition 6.17 *MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$ are* independent*, denoted $\mathcal{M}_1 \ddagger \mathcal{M}_2$, iff every action of $\mathcal{M}_1$ is independent of every action of $\mathcal{M}_2$ in $\mathcal{M}_1||\mathcal{M}_2$.*

Let $s \xrightarrow{*} s'$ denote that state s' is reachable from s through an arbitrary finite sequence of transitions in MTS $\mathcal{M}$. In other words, $s \xrightarrow{*} s'$ means that there exists a finite execution ρ in $\mathcal{M}$ that starts in state s and ends in s'.

Definition 6.18 (Layered composition) *Given MTSs $\mathcal{M}_i = (S_i, Act_i, s_{0i}, \{s_{fi}\}, V_i)$, where $i \in \{1,2\}$ with $S_1 \cap S_2 = \varnothing$. The* layered composition *of $\mathcal{M}_1$ and $\mathcal{M}_2$, denoted $\mathcal{M}_1 \bullet \mathcal{M}_2$, is the MTS $(S_1 \times S_2, Act_1 \cup Act_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, V)$ where for all $(s, s') \in S_1 \times S_2$, V is defined by:*

- *For all $a \in Act_1 \cap Act_2$, if there exists $u \in S_1$ and $u' \in S_2$, such that $V_1(s,a,u) \neq \bot$ and $V_2(s',a,u') \neq \bot$, define $V((s,s'),a,(u,u')) = V_1(s,a,u) \sqcap V_2(s',a,u')$. If either $\forall u \in S_1$, we have $V_1(s,a,u) = \bot$, or $\forall u' \in S_2$, we have $V_2(s',a,u') = \bot$ then $\forall (u,u') \in S_1 \times S_2$, $V((s,s'),a,(u,u')) = \bot$.*
- *For all $(u,u') \in S_1 \times S_2$, $a \in Act_1 \setminus Act_2$, define $V((s,s'),a,(u,u')) = V_1(s,a,u)$ if $s' = u'$, $V((s,s'),a,(u,u')) = \bot$ otherwise.*
- *For all $(u,u') \in S_1 \times S_2$, $a \in Act_2 \setminus Act_1$, define $V((s,s'),a,(u,u')) = V_2(s',a,u')$ if $s = u \wedge \forall s* \; s \xrightarrow{*} s* : act(s*) \ddagger a$, $V((s,s'),a,(u,u')) = \bot$ otherwise.*

Note that the first two clauses of Def. 6.18 are the same as for Def. 6.14. The crux of the definition of layered composition lies in the last clause. This clause defines V in the same way as for parallel composition, in case the action in $\mathcal{N}_2$ is independent of all actions enabled in states reachable from the current state in $\mathcal{N}_1$. In case an action a' in $\mathcal{N}_2$ depends on one or more actions in $\mathcal{N}_1$, then this clause ensures that it cannot be executed before all the actions in $\mathcal{N}_1$ (on which it depends) have taken place. In other words, all finite executions, in which d is executed before any action say a, such that $a \dagger d$ will not be part of $Exec_{fin}(\mathcal{M}_1 \bullet \mathcal{M}_2)$.

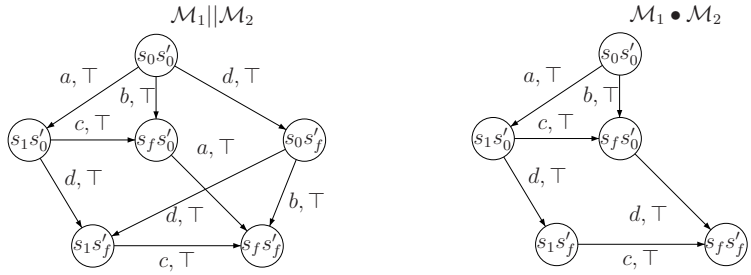

Figure 6.4: Parallel composition $\mathcal{M}_1||\mathcal{M}_2$ (left) and layered composition $\mathcal{M}_1 \bullet \mathcal{M}_2$ (right) where $a \dagger d$

Example 6.19 *Let us assume that actions a, d are dependent in $\mathcal{M}_1||\mathcal{M}_2$ (Fig. 6.4 (left)). Then the layered composition $\mathcal{M}_1 \bullet \mathcal{M}_2$ is shown in Fig. 6.4 (right). Note that the execution in which d is executed before a has been removed in $\mathcal{M}_1 \bullet \mathcal{M}_2$.*

Next, we use the above mentioned composition operators for formulating the communication closed layer (CCL) laws as follows:

Theorem 6.20 (CCL laws) *For MTSs $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{M}_1$, and $\mathcal{M}_2$, with $\mathcal{N}_1 \ddagger \mathcal{M}_2$ and $\mathcal{M}_1 \ddagger \mathcal{N}_2$, the following communication closed layer (CCL) equivalences hold:*

1. $\mathcal{N}_1 \bullet \mathcal{M}_2 \equiv \mathcal{N}_1||\mathcal{M}_2$ *(IND)*
2. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_2 \equiv \mathcal{N}_1 \bullet (\mathcal{N}_2||\mathcal{M}_2)$ *(CCL-L)*
3. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_1 \equiv (\mathcal{N}_1||\mathcal{M}_1) \bullet \mathcal{N}_2$ *(CCL-R)*
4. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||(\mathcal{M}_1 \bullet \mathcal{M}_2) \equiv (\mathcal{N}_1||\mathcal{M}_1) \bullet (\mathcal{N}_2||\mathcal{M}_2)$ *(CCL)*

Intuitively, CCL laws allow moving the $\bullet$ operator out of brackets such that the two systems are refinement equivalent. These laws are required for carrying out the structural transformations.

6.3 Partial Order Equivalence and Property Preservation

This section defines the notion of partial order equivalence ($\equiv^*_{po}$) between MTSs which is used to prove that sequential and layered composition of MTSs satisfy

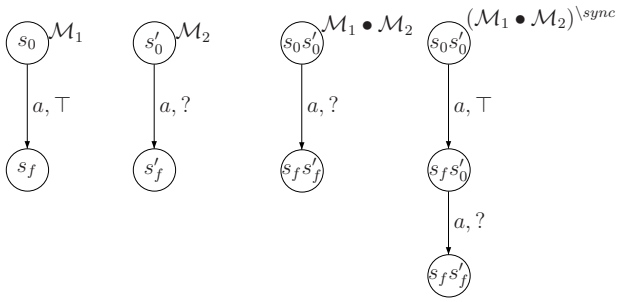

Figure 6.5: MTS without synchronized transitions

the same existential ($\exists$) and universal ($\forall$) reachability properties. For MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$, let $\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}_2$. We define $\mathcal{M}^{\setminus sync}$ as the MTS obtained from $\mathcal{M}$ such that it does not have any synchronized transitions (which are present in $\mathcal{M}$ as a result of synchronization over common actions). Intuitively, this means that executions in $\mathcal{M}$ with synchronized transitions and ending in some final state can be rewritten in $\mathcal{M}^{\setminus sync}$ such that for every execution there are corresponding executions in $\mathcal{M}^{\setminus sync}$ obtained by allowing interleaving on common actions. Note that for every common action, we only allow interleaving (in $\mathcal{M}^{\setminus sync}$) from the state of $\mathcal{M}$ where the synchronization over that action takes place. For example, let $\mathcal{M}$ have only one transition (a may a-transition) which is a result of synchronization of a must a-transition (from $\mathcal{M}_1$), and a may a- transition (from $\mathcal{M}_2$). In this case $\mathcal{M}^{\setminus sync}$ will have a corresponding[1] sequence of transitions, i.e., a must a-transition (from $\mathcal{M}_1$) followed by a may a-transition (from $\mathcal{M}_2$). This transformation is required as we want to establish the result that layered composition which incorporates synchronization is po-equivalent to sequential composition which does not incorporate synchronization. This means that for any $\mathcal{M} = \mathcal{M}_1; \mathcal{M}_2$, it holds $\mathcal{M}^{\setminus sync} = \mathcal{M}$.

Example 6.21 *Consider the MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$ shown in Fig. 6.5 (left). The layered composition of $\mathcal{M}_1, \mathcal{M}_2$, i.e., $\mathcal{M}_1 \bullet \mathcal{M}_2$ is shown in the middle, where $\mathcal{M}_1$ and $\mathcal{M}_2$ synchronize on common action a. A may transition is obtained in $\mathcal{M}_1 \bullet \mathcal{M}_2$ (since composing must-may results in a may transition). An MTS $(\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync}$ without synchronized transitions is shown in Fig. 6.5 (right). Here the common action of $\mathcal{M}_1$ is executed before the common action of $\mathcal{M}_2$.*

[1] In fact, if we would not restrict ourselves to deterministic MTSs, then two corresponding transition sequences making a diamond shape would be obtained in $\mathcal{M}^{\setminus sync}$.

Theorem 6.22 *For MTSs $\mathcal{M}_1$ and $\mathcal{M}_2$, let $\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}_2$, and S_f be the set of final states in $\mathcal{M}$. Then the following holds:*

$$\mathcal{M} \models^{\exists} \Diamond S_f \Leftrightarrow \mathcal{M}^{\setminus sync} \models^{\exists} \Diamond S_f$$

$$\mathcal{M} \models^{\forall} \Diamond S_f \Leftrightarrow \mathcal{M}^{\setminus sync} \models^{\forall} \Diamond S_f$$

In simple words this theorem says that reasoning about $\mathcal{M}^{\setminus sync}$ in place of $\mathcal{M}$ is not a restriction as the behaviour of $\mathcal{M}$ (w.r.t. reachability properties) is completely mimicked by $\mathcal{M}^{\setminus sync}$. Next, we define the notion of partial order equivalence between two finite executions.

Definition 6.23 (po-equivalence) *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MTSs with transition functions V_1 and V_2, respectively. Let $\rho_1 \in Exec_{fin}(\mathcal{M}_1^{\setminus sync})$ and $\rho_2 \in Exec_{fin}(\mathcal{M}_2^{\setminus sync})$. Then $\rho_1 \equiv_{po} \rho_2$ iff there exist finite executions ρ', ρ'' and $\exists a_1, b_1$ with $a_1 \neq b_1$ such that the following holds:*

- $\forall i < |\rho'|$: $V_1(s_i, a_{i+1}, s_{i+1}) = V_2(s_i, a_{i+1}, s_{i+1})$.
- $\forall (|\rho'|+2) \leq i < (|\rho'|+|\rho''|+2) : V_1(s_i, a_{i+1}, s_{i+1}) = V_2(s_i, a_{i+1}, s_{i+1})$.
- $\rho_1 = \rho' a_1 s b_1 \rho'' \wedge \rho_2 = \rho' b_1 s' a_1 \rho''$, *where* $a_1 \ddagger b_1$.
- $V_1(last(\rho'), a_1, s) = V_2(s', a_1, first(\rho'')) \wedge V_1(s, b_1, first(\rho'')) = V_2(last(\rho'), b_1, s')$.

Let $\equiv_{po}^{*}$ called po-equivalence *denote the reflexive, transitive closure of $\equiv_{po}$.*

In simple words, if two finite executions ρ_1, ρ_2 are po-equivalent, then ρ_1 can be obtained from ρ_2 by repeated permutation of adjacent independent actions. Note that the first two conditions of Def. 6.23 are required to ensure that if for example $\rho' = s_0 c_1 s_1 d_1 s_2$ where c_1 is a must transition and d_1 is a may transition in $\mathcal{M}_1^{\setminus sync}$, then c_1, d_1 are also must and may transitions in $\mathcal{M}_2^{\setminus sync}$. Let $\mathcal{M}_1 \circ \mathcal{M}_2$ denote $(\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync}$.

Definition 6.24 (LNF) *Let S_f be the set of final states in $\mathcal{M}_1 \circ \mathcal{M}_2$. Then $\rho \in Exec_{fin}^{S_f}(\mathcal{M}_1 \circ \mathcal{M}_2)$ is in* layered normal form *(LNF) iff it involves the consecutive execution of actions of $\mathcal{M}_1$, followed by the consecutive execution of actions of $\mathcal{M}_2$.*

Let $Exec^{LNF}_{fin}(\mathcal{M}_1 \circ \mathcal{M}_2)$ denote the set of all finite executions in $Exec^{S_f}_{fin}(\mathcal{M}_1 \circ \mathcal{M}_2)$ that are in LNF.

Next we show that for each finite execution of $Exec^{S_f}_{fin}(\mathcal{M}_1 \circ \mathcal{M}_2)$, a po-equivalent exceution in LNF does exist.

Lemma 6.25 (LNF existence) *Let $\mathcal{M}_1, \mathcal{M}_2$ be two MTSs. Then we have $\forall \rho \in Exec^{S_f}_{fin}(\mathcal{M}_1 \circ \mathcal{M}_2) \exists \rho' \in Exec^{LNF}_{fin}(\mathcal{M}_1 \circ \mathcal{M}_2)$ such that $\rho \equiv^*_{po} \rho'$.*

Definition 6.26 (po-equivalence for MTSs) *Two MTSs $\mathcal{M}_1, \mathcal{M}_2$ are said to be po-equivalent, denoted $\mathcal{M}_1 \equiv^*_{po} \mathcal{M}_2$, iff for $i \in \{1,2\}$: $\forall \rho_i \in Exec^{S_{fi}}_{fin}(\mathcal{M}_i^{\setminus sync}) \exists \rho_{3-i} \in Exec^{S_{f3-i}}_{fin}(\mathcal{M}_{3-i}^{\setminus sync})$ such that $\rho_i \equiv^*_{po} \rho_{3-i}$.*

Theorem 6.27 *For any two MTSs $\mathcal{M}_1, \mathcal{M}_2$, it holds: $\mathcal{M}_1 \bullet \mathcal{M}_2 \equiv^*_{po} \mathcal{M}_1; \mathcal{M}_2$.*

In simple words, this theorem says that for each finite execution in layered composition of two MTSs, a po-equivalent execution in the their sequential composition does exist and vice versa. This result can be used to replace ; by $\bullet$ provided po-equivalence preserves reachability properties.

Example 6.28 *It is easy to check that MTS $\mathcal{M}_1; \mathcal{M}_2$ given in Fig. 6.3 (right) is po-equivalent to MTS $\mathcal{M}_1 \bullet \mathcal{M}_2$ given in Fig. 6.4 (right).*

Theorem 6.29 (Property preservation) *Let $\mathcal{M}, \mathcal{M}'$ be two MTSs with set of final states S_f, and S'_f, respectively. If $\mathcal{M} \equiv^*_{po} \mathcal{M}'$ then we have:*

$$\mathcal{M} \models^{\exists} \Diamond S_f \text{ iff } \mathcal{M}' \models^{\exists} \Diamond S'_f$$

$$\mathcal{M} \models^{\forall} \Diamond S_f \text{ iff } \mathcal{M}' \models^{\forall} \Diamond S'_f$$

This theorem asserts that po-equivalent MTSs satisfy the same reachability properties.

Proposition 6.30 *Let $\mathcal{M}$, $\mathcal{M}'$ and $\mathcal{M}_1$ be three MTSs such that $\mathcal{M} \equiv^*_{po} \mathcal{M}'$. Then we have:*

$$\mathcal{M} || \mathcal{M}_1 \equiv^*_{po} \mathcal{M}' || \mathcal{M}_1$$

The results of Theorem 6.27, Theorem 6.29 and Proposition 6.30 enable us to replace ; with $\bullet$ and vice versa. This replacement along with CCL laws (Theorem 6.20) can be used for state space reduction as follows:

State space reduction. Let $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_1$ be three MTSs, and $\mathcal{N} = (\mathcal{N}_1; \mathcal{N}_2) || \mathcal{M}_1$. Let us say we want to check whether $\mathcal{N} \models^{\exists} \Diamond S_f$ or $\mathcal{N} \models^{\forall} \Diamond S_f$

where S_f is the set of final states in $\mathcal{N}$. Assume $\mathcal{M}_1 \ddagger \mathcal{N}_2$, and $\mathcal{N}_1, \mathcal{N}_2, \mathcal{M}_1$ each consist of 20 states. In this case $\mathcal{N}_1; \mathcal{N}_2$ has 39 states which combined with the 20 states of $\mathcal{M}_1$ gives 780 states. The results in this chapter, allow for transforming $\mathcal{N}$ in the following way:

$$
\begin{array}{ll}
\underbrace{(\mathcal{N}_1; \mathcal{N}_2)||\mathcal{M}_1}_{\mathcal{N}} & \\
\equiv^*_{po} & \text{Theorem 6.27, Proposition 6.30} \\
(\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_1 & \\
\equiv & \text{Theorem 6.20 (CCL-R)} \\
(\mathcal{N}_1||\mathcal{M}_1) \bullet \mathcal{N}_2 & \\
\equiv^*_{po} & \text{Theorem 6.27} \\
(\mathcal{N}_1||\mathcal{M}_1); \mathcal{N}_2 &
\end{array}
$$

Note that the transformed system, i.e., $(\mathcal{N}_1||\mathcal{M}_1); \mathcal{N}_2$ has 419 states.

6.4 Possible Extensions

In this section we briefly discuss the extension of our results to acyclic MTSs equipped with data variables.

MTS with data. An MTS $\mathcal{M}$ can be extended with data variables like in [13] such that whenever an action is executed its associated data variables are updated according to an assignment. These data variables can take values in some finite range D. The definitions of satisfaction, and refinement can be slightly modified by imposing an extra condition that ensures that related states have the same data valuations. As mentioned earlier, for an MTS with data, two actions are said to be dependent if one of the two writes a variable that is read or written by the other action. Using this dependency relation, our theory can be applied to acyclic MTSs with data. We do not go into details on these matters here, however, refer the interested reader to [13, 82, 149].

6.5 Related Work

The decomposition of a distributed program into communication closed layers to simplify its analysis was originally proposed in [51]. In [83], a layered composition operator and various algebraic transformation rules have been introduced to simplify the analysis of distributed database systems. Some other examples where

layering techniques have been applied for the analysis of distributed systems can be found in [82, 84, 85]. In [84], layering techniques have been applied to derive the implementation of a distributed minimum weight spanning tree algorithm. An extension of layering and CCL laws to the real-time setting has been proposed in [85]. This technique has been successfully applied to obtain a simpler proof of correctness for Fischer's real time mutual exclusion algorithm. Layered composition for timed automata (TA) has been investigated in [123]. In [123], the usefulness of layering based state space reduction for a network of timed automata has been illustrated by considering a collision avoidance protocol developed for an audio/video system. Recently, layering based structural transformations for TA have been applied for easier verification of Fischer's real-time mutual exclusion protocol [124].

6.6 Conclusions

This chapter presented a state-space reduction technique for a network of MTSs, based on the notion of layering. We proposed a layered composition operator, and formulated communication closed layer (CCL) laws. Next, we defined the partial order (po) equivalence between MTSs, proved that layered and sequential compostion operators are po-equivalent and satisfy the same existential ($\exists$) and universal ($\forall$) reachability properties. Finally, we discussed the possible extensions of our results. As implementations of distributed systems typically are in terms of layers, we believe that enabling transforming system MTS specifications into layered form will substantially ease the proof of correct implementation.

6. LAYERED REDUCTION FOR MODAL SPECIFICATION THEORIES

Chapter 7

Layered Reduction for Abstract Probabilistic Automata

In the previous chapter, we have seen that layering based structural transformations can be used for reducing the state space of a network of acyclic MTSs. This chapter proposes a complete theory of layered reduction for a network of acyclic abstract probabilistic automata (APAs). APAs provide a powerful abstraction and specification formalism for sets of PAs, support compositionality and a stepwise refinement methodology. They are thus suitable for component-oriented design and analysis of randomized distributed algorithms. As discussed in chapter 6, action dependencies between sub-components can be either explicitly stated or derived from the operations performed on data variables that are updated during an action execution (in case of APA with data). Based on the notion of action independence we propose a layered composition operator and use it to formulate and prove Commuication Closed Layer (CCL) laws for acyclic APAs. We also propose a partial order relation which is required to relate sequential composition operator and layered composition operator. Next, we focus on proving that the reduced system preserves maximum (resp. minimum) probability to reach its set of final states. As a result, probabilistic reachability properties can be checked on the layered, typically smaller, model.

Contributions. More specifically, the main contributions of this chapter are as follows:

- We define the notions of *abstract execution* and *realisation*, which are subsequently used to compare the behaviour of acyclic APAs.
- We define the layered ($\bullet$) and sequential ($;$) composition operator, and

formulate communication closed layer (CCL) laws for acyclic APAs.

- We define the partial order (po) equivalence between acyclic APAs, show that $\bullet$ is po-equivalent to ;, and prove that po-equivalent APAs have the same maximum (resp. minimum) probabilities to reach the set of final states.
- Finally, we show how state space reduction can be achieved by replacing $\bullet$ with ; within the framework of CCL laws.

Organisation of this chapter. Section 7.1 presents the satisfaction and refinement relations for APAs. Section 7.2 discusses the composition operators for APAs, and introduces CCL laws. Section 7.3 defines po-equivalence between APAs, and proves that po-equivalence between $\bullet$ and ; preserves maximum (resp. minimum) reachability probabilities. Section 7.4 discusses the possible extensions of our results. Section 7.5 discusses related work. Finally, Section 7.6 concludes the chapter.

7.1 Satisfaction and Refinement

This section presents the notions of *satisfaction* and *refinement* originally introduced in [44]. A satisfaction relation allows to relate a PA (implementation) with an APA (specification). A refinement relation is used to compare APAs w.r.t. their sets of implementations. We also define the notions of realisation and maximum (resp. minimum) probabilities of reaching the set of final states computed over all the implementations of an APA.

Definition 7.1 (Simulation relation) *Let S and S' be non-empty sets of states. Given $\mu \in Dist(S)$, $\mu' \in Dist(S')$, a function $\delta : S \to (S' \to [0,1])$, and a binary relation $R \subseteq S \times S'$, μ is* simulated *by μ' with respect to R and δ, denoted as $\mu \Subset^{\delta}_{R} \mu'$, iff*

- *for all $s \in S$, if $\mu(s) > 0$, then $\delta(s) \in Dist(S')$,*
- *for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and*
- *if $\delta(s)(s') > 0$, then $(s, s') \in R$.*

We write $\mu \Subset_R \mu'$ iff there exists a function δ such that $\mu \Subset^{\delta}_{R} \mu'$. Such δ is called a correspondence *function.*

Definition 7.2 (Satisfaction [44]) *Let $\mathcal{P} = (S, s_0, S_f, Act, V)$ be a PA and $\mathcal{N} = (S', s'_0, S'_f, Act, V')$ be an APA. $R \subseteq S \times S'$ is a* satisfaction relation *iff, for any $(s, s') \in R$, the following conditions hold:*

- $\forall a \in Act, \forall \varphi' \in C(S') : V'(s', a, \varphi') = \top \Rightarrow \exists \mu \in Dist(S) : V(s, a, \mu) = \top$ *and* $\exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$,
- $\forall a \in Act, \forall \mu \in Dist(S) : V(s, a, \mu) = \top \Rightarrow \exists \varphi' \in C(S') : V'(s', a, \varphi') \neq \bot$ *and* $\exists \mu' \in Sat(\varphi')$ *such that* $\mu \Subset_R \mu'$,
- $s \in S_f \Leftrightarrow s' \in S'_f$.

We say that $\mathcal{P}$ satisfies *$\mathcal{N}$, denoted $\mathcal{P} \models \mathcal{N}$, iff there exists a satisfaction relation relating s_0 and s'_0. If $\mathcal{P} \models \mathcal{N}$, $\mathcal{P}$ is called an* implementation *of $\mathcal{N}$. Let $[\![\mathcal{N}]\!] = \{\mathcal{P} \mid \mathcal{P} \models \mathcal{N}\}$, the set of implementations of $\mathcal{N}$.*

Intuitively, a state s in PA $\mathcal{P}$ satisfies state s' in APA $\mathcal{N}$ iff any must transition of s' is matched by a transition of s agreeing on the probability distributions specified by the constraint, and s does not contain any transitions without a corresponding transition (may or must) in s'. In addition, final states in $\mathcal{P}$ can only be related to final states in $\mathcal{N}$ and vice versa.

Definition 7.3 (Realisation) *Let $\mathcal{P} \in [\![\mathcal{N}]\!]$, and $\pi = s'_0 a'_1 \mu'_1 s'_1 a'_2 \mu'_2 s'_2 \ldots s'_n$ be a finite path of $\mathcal{P}$, i.e., $\pi \in Paths_{fin}(\mathcal{P})$. π is said to be a* realisation *of $\rho = s_0 a_1 \varphi_1 s_1 a_2 \varphi_2 s_2 \ldots s_n$ where $\rho \in Exec_{fin}(\mathcal{N})$, denoted $\pi \models \rho$, if $\forall i < n : \mu'_{i+1} \in Sat(\varphi_{i+1}) \wedge a'_{i+1} = a_{i+1}$.*

Definition 7.4 (Realisable) *An implementation $\mathcal{P} \in [\![\mathcal{N}]\!]$ is said to be* realisable, *if $\mathcal{P}$ is deterministic and $\forall \pi \in Paths_{fin}(\mathcal{P}) \exists \rho \in Exec_{fin}(\mathcal{N}) : \pi \models \rho$.*

Example 7.5 *The PA $\mathcal{P}$ in Fig. 7.1 (right) is an implementation of the APA $\mathcal{N}$ in Fig. 7.1 (left). It is easy to check that there exists a satisfaction relation relating initial states of $\mathcal{P}$ and $\mathcal{N}$. Note that in this example, for every implementation of $\mathcal{N}$, $S_f \neq \varnothing$ (since the value of x_1 is always greater than 0 and s_2 has a must-b transition moving to s_f with probability 1). It can be checked that $\mathcal{P}$ is a realisable implementation of $\mathcal{N}$ (since $\mathcal{P}$ is deterministic and every finite path π of $\mathcal{P}$ is a realisation of some finite abstract execution ρ in $\mathcal{N}$).*

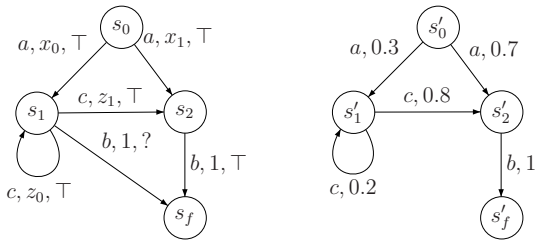

Figure 7.1: An APA $\mathcal{N}$ (left) and a PA $\mathcal{P}$ (right) that satisfies $\mathcal{N}$

Example 7.6 *Finite path* $\pi = s'_0 a \mu'_1 s_1 c \mu'_2 s_1 c \mu'_2 s'_2$ *of PA* $\mathcal{P}$ *(Fig. 7.1 (right)) is a realisation of finite abstract execution* $\rho = s_0 a \varphi_1 s_1 c \varphi_2 s_1 c \varphi_2 s_2$ *of* $\mathcal{N}$ *(Fig. 7.1 (left)), where* $\mu'_1(s_1) = 0.3, \mu'_1(s_2) = 0.7, \mu'_2(s_1) = 0.2, \mu'_2(s_2) = 0.8$, *and* $\mu'_3(s_f) = 1$.

From the definition of satisfaction and realisation it can be checked that for an APA $\mathcal{N}$, there can be implementations where none of the finite paths is a realisation of some finite abstract execution of $\mathcal{N}$. Note that for a deterministic APA $\mathcal{N}$ with $\mathcal{P} \models \mathcal{N}$, if a path $\pi \in Paths_{fin}(\mathcal{P})$ is a realisation of $\rho \in Exec_{fin}(\mathcal{N})$, then π cannot be a realisation of another finite abstract execution of $\mathcal{N}$.

Definition 7.7 (Refinement) *Let* $\mathcal{N} = (S, s_0, S_f, Act, V)$ *and* $\mathcal{N}' = (S', s'_0, S'_f, Act, V')$ *be APAs.* $R \subseteq S \times S'$ *is a strong* refinement relation *iff, for all* $(s, s') \in R$, *the following conditions hold:*

- $\forall a \in Act. \forall \varphi' \in C(S'). V'(s', a, \varphi') = \top \Rightarrow \exists \varphi \in C(S). V(s, a, \varphi) = \top$ *and there exists a correspondence function* $\delta : S \to (S' \to [0,1])$ *such that* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset^{\delta}_{R} \mu'$,
- $\forall a \in Act. \forall \varphi \in C(S). V(s, a, \varphi) \neq \bot \Rightarrow \exists \varphi' \in C(S'). V'(s', a, \varphi') \neq \bot$ *and there exists a correspondence function* $\delta : S \to (S' \to [0,1])$ *such that* $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ *with* $\mu \Subset^{\delta}_{R} \mu'$.
- $s \in S_f \Leftrightarrow s' \in S'_f$.

$\mathcal{N}$ *strongly refines* $\mathcal{N}'$, *denoted* $\mathcal{N} \preceq_S \mathcal{N}'$, *iff there exists a strong refinement relation relating* s_0 *and* s'_0.

Intuitively, a state s strongly refines state s' iff any must transition of s' is matched by a transition of s agreeing on the probability distributions specified by the constraint, and s does not contain any transitions (may or must) without a corresponding transition (may or must) in s'. In addition, the final states in $\mathcal{N}$ can only be related to final states in $\mathcal{N}'$ and vice versa.

Definition 7.8 (Refinement equivalence) *APAs $\mathcal{N}$ and $\mathcal{N}'$ are* refinement equivalent*, denoted $\mathcal{N} \equiv \mathcal{N}'$, iff $\mathcal{N} \preceq_S \mathcal{N}'$ and $\mathcal{N}' \preceq_S \mathcal{N}$.*

Since strong refinement implies inclusion of sets of implementations, if $\mathcal{N} \equiv \mathcal{N}'$, then they have the same set of implementations, i.e., $[\![\mathcal{N}]\!] = [\![\mathcal{N}']\!]$ [44].

Remark 7.9 *A satisfaction relation can be seen as a special case of refinement relation. In simple words, if $\mathcal{P} \models \mathcal{N}$, then $\mathcal{P} \preceq_S \mathcal{N}$ (since every PA is an APA and all the three conditions of strong refinement are satisfied).*

Similar to the case of layering for MTSs (Chapter 6), we make the following assumptions:

- Every APA is acyclic and consistent. An APA is consistent iff it admits at least one implementation. See [44] for more details.
- Every APA has a single final state, i.e., $|S_f|= 1$, and all its states (except the final state) have at least one outgoing transition.
- Dependencies between actions of different components are known in advance.

In this chapter, we focus on *maximum reachability probabilities* (resp. *minimum*) of reaching the set of final states $S_f \subset S$ in a PA, over all possible adversaries. Maximum reachability probabilities are defined as follows:

$$P_{\mathcal{P}}^{max}(S_f) = \sup_{\mathcal{D} \in Adv(\mathcal{P})} P^{\mathcal{D}}(S_f)$$

$$P^{\mathcal{D}}(S_f) = \sum_{\pi \in Paths_{fin}^{\mathcal{D},S_f}(\mathcal{P})} P^{\mathcal{D}}(\pi)$$

Minimum reachability probabilities are defined in an analogous manner.

Next, we define the maximum reachability probabilities (resp. minimum) of reaching the set of final states, determined over all the implementations of an APA $\mathcal{N}$. Intuitively this means that for each implementation $\mathcal{P} \in [\![\mathcal{N}]\!]$, we are interested in computing the maximum (resp. minimum) probability to reach the set of states S_f, and finally we would compute the maximum (resp. minimum) probability over all the implementations. More formally it is defined as follows:

$$P_{\mathcal{N}}^{max}(S_f) = \sup_{\mathcal{P} \in [\![\mathcal{N}]\!]} P_{\mathcal{P}}^{max}(S_f)$$

Minimum reachability probabilities are defined in an analogous manner.

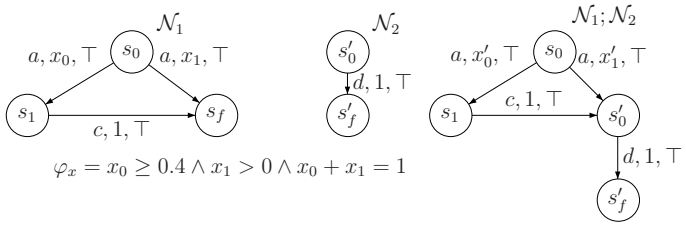

Figure 7.2: APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ (left) and their sequential composition (right)

7.2 Composition and CCL Laws

In this section we define composition operators that enable to combine two APAs. We propose sequential, and layered composition operators, and recall parallel composition from [44]. The framework of CCL laws is also formulated, which is required for carrying out the layered transformations.

Definition 7.10 (Sequential composition) *Given APAs* $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$*, where* $i \in \{1, 2\}$ *with* $S_1 \cap S_2 = \varnothing$*. Their* sequential composition*, denoted* $\mathcal{N}_1;\mathcal{N}_2$*, is the APA* $(S, s_{01}, \{s_{f2}\}, Act_1 \cup Act_2, V)$*, where* $S = S_1 \backslash \{s_{f1}\} \cup S_2$ *and* $V = V'_1 \cup V_2$*. Here* $V'_1 = V_1[s_{02} \leftarrow s_{f1}]$ *is defined by* $V'_1(s, a, \varphi') = V_1(s, a, \varphi)$ *with* φ' *the new constraint in* $C(S)$ *such that for any* μ*,* $\mu \in Sat(\varphi)$ *iff there exists* $\mu' \in Sat(\varphi')$ *with*

$$\mu'(s') = \mu(s') \text{ if } s' \neq s_{f1}, \text{ and}$$
$$\mu'(s_{02}) = \mu(s_{f1}) \text{ otherwise.}$$

Intuitively, the sequential composition of two APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ requires that the actions of $\mathcal{N}_2$ can only be executed once the final state of $\mathcal{N}_1$, i.e., s_f has been reached. Note that all the incoming transitions to state s_{f1} are redirected to s_{02}. Here, s_{01}, s_{f2} are the new initial and final states in the resulting APA $\mathcal{N}_1;\mathcal{N}_2$, respectively.

Definition 7.11 (Parallel composition) *Given APAs* $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$*, where* $i \in \{1, 2\}$ *with* $S_1 \cap S_2 = \varnothing$*. Their* parallel composition*, denoted* $\mathcal{N}_1 || \mathcal{N}_2$*, is the APA* $(S_1 \times S_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, Act_1 \cup Act_2, V)$ *where for all* $(s, s') \in S_1 \times S_2$*,* V *is defined by:*

- *For all* $a \in Act_1 \cap Act_2$*, if there exists* $\varphi \in C(S_1)$ *and* $\varphi' \in C(S_2)$*, such that* $V_1(s, a, \varphi) \neq \bot$ *and* $V_2(s', a, \varphi') \neq \bot$*, then* $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi) \sqcap V_2(s', a, \varphi')$ *with* $\tilde{\varphi}$ *the new constraint in* $C(S_1 \times S_2)$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff there exists* $\mu \in Sat(\varphi)$ *and* $\mu' \in Sat(\varphi')$ *such that* $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$ *for all* $u \in S_1$ *and* $v \in S_2$*. If either for all* $\varphi \in C(S_1)$*, we have* $V_1(s, a, \varphi) = \bot$*,*

or $\forall \varphi' \in C(S_2)$*, we have* $V_2(s', a, \varphi') = \bot$ *then for all* $\tilde{\varphi} \in C(S_1 \times S_2)$*,* $V((s, s'), a, \tilde{\varphi}) = \bot$*.*

- *For all* $a \in Act_1 \setminus Act_2$*, and for all* $\varphi \in C(S_1)$*, define* $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi)$ *with* $\tilde{\varphi}$ *the new constraint in* $C(S_1 \times S_2)$ *such that* $\tilde{\mu} \in Sat(\tilde{\varphi})$ *iff for all* $u \in S_1$ *and* $v \neq s', \tilde{\mu}(u, v) = 0$ *and the distribution* $\mu : t \mapsto \tilde{\mu}(t, s')$ *is in* $Sat(\varphi)$*.*

- *For all* $a' \in Act_2 \setminus Act_1$ *and for all* $\varphi' \in C(S_2)$*, define* $V((s, s'), a', \tilde{\varphi}') = V_2(s', a', \varphi')$ *with* $\tilde{\varphi}'$ *the new constraint in* $C(S_1 \times S_2)$ *such that* $\tilde{\mu}' \in Sat(\tilde{\varphi}')$ *iff for all* $u \neq s$ *and* $v \in S_2, \tilde{\mu}'(u, v) = 0$ *and the distribution* $\mu' : t' \mapsto \tilde{\mu}'(s, t')$ *is in* $Sat(\varphi')$*.*

Parallel composition forces synchronization on all common actions and interleaving on disjoint actions. The synchronization of two must transitions results in a must transition, and composing may-must, must-may and may-may transitions results in a may transition. Note that $||$ is commutative and associative.

Next, we introduce the notion of action independence which is subsequently used to define layered composition. The dependency between actions a and b is denoted $a \dagger b$. Two additional requirements for the dependency relation are that it is reflexive and symmetric. Two distinct actions that are not dependent are said to be independent, where independence is defined as follows:

Definition 7.12 *Let APA* $\mathcal{N} = (S, s_0, S_f, Act, V)$*. Actions* $a, b \in Act$ *such that* $a \neq b$ *are* independent *in* $\mathcal{N}$*, denoted* $a \ddagger b$*, iff for all states* $s \in S$ *with* $a, b \in act(s)$ *we have:*

- *For any* $s' \in S$*:* $s' \in (a, \varphi)(s) \Rightarrow b \in act(s')$*,*
- *For any* $s' \in S$*:* $s' \in (b, \varphi')(s) \Rightarrow a \in act(s')$*, and*
- *For any* $s'' \in S$*:*
  - $\forall \rho \in Exec^s_{fin}(\mathcal{N}) : \rho = sa\varphi s'b\varphi' s'' \Rightarrow \exists \rho' \in Exec^s_{fin}(\mathcal{N}) : \rho' = sb\varphi' s'a\varphi s''$*.*
  - $\forall \rho \in Exec^s_{fin}(\mathcal{N}) : \rho = sb\varphi' s'a\varphi s'' \Rightarrow \exists \rho' \in Exec^s_{fin}(\mathcal{N}) : \rho' = sa\varphi s'b\varphi' s''$*.*

The first two conditions assert that a and b should not disable each other. The last condition asserts that the same state should be reached from s in two steps

by either performing a followed by b, or by performing b followed by a. Let $act(s) \ddagger a$ denote $\forall b \in act(s) : b \ddagger a$. Note that this notion of action independence is a purely syntactic notion, and —in contrast to action independence in partial-order reduction [7]— does not take the transition probabilities into account.

Definition 7.13 *APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ are* independent*, denoted $\mathcal{N}_1 \ddagger \mathcal{N}_2$, iff every action of $\mathcal{N}_1$ is independent of every action of $\mathcal{N}_2$ in $\mathcal{N}_1||\mathcal{N}_2$.*

Let $s \xrightarrow{*} s'$ denote that state s' is reachable from s through an arbitrary finite sequence of transitions in APA $\mathcal{N}$. In other words, $s \xrightarrow{*} s'$ denote that there exists a finite abstract execution $\rho \in Exec^s_{fin}(\mathcal{N})$ with $last(\rho) = s'$.

Definition 7.14 (Layered composition) *Given APAs $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$, where $i \in \{1, 2\}$ with $S_1 \cap S_2 = \varnothing$. Their* layered composition*, denoted $\mathcal{N}_1 \bullet \mathcal{N}_2$, is the APA $(S_1 \times S_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, Act_1 \cup Act_2, V)$ where for all $(s, s') \in S_1 \times S_2$, V is defined by:*

- *For all $a \in Act_1 \cap Act_2$, if there exists $\varphi \in C(S_1)$ and $\varphi' \in C(S_2)$, such that $V_1(s, a, \varphi) \neq \bot$ and $V_2(s', a, \varphi') \neq \bot$, then $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi) \sqcap V_2(s', a, \varphi')$ with $\tilde{\varphi}$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff there exists $\mu \in Sat(\varphi)$ and $\mu' \in Sat(\varphi')$ such that $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$ for all $u \in S_1$ and $v \in S_2$. If either for all $\varphi \in C(S_1)$, we have $V_1(s, a, \varphi) = \bot$, or $\forall \varphi' \in C(S_2)$, we have $V_2(s', a, \varphi') = \bot$ then for all $\tilde{\varphi} \in C(S_1 \times S_2)$, $V((s, s'), a, \tilde{\varphi}) = \bot$.*

- *For all $a \in Act_1 \setminus Act_2$, and for all $\varphi \in C(S_1)$, define $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi)$ with $\tilde{\varphi}$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff for all $u \in S_1$ and $v \neq s', \tilde{\mu}(u, v) = 0$ and the distribution $\mu : t \mapsto \tilde{\mu}(t, s')$ is in $Sat(\varphi)$.*

- *For all $a' \in Act_2 \setminus Act_1$ and for all $\varphi' \in C(S_2)$:*

  1. *If $\forall s^* : s \xrightarrow{*} s* : act(s^*) \ddagger a'$ in $\mathcal{N}_1||\mathcal{N}_2$, then we define $V((s, s'), a', \tilde{\varphi}') = V_2(s', a', \varphi')$ with $\tilde{\varphi}'$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu}' \in Sat(\tilde{\varphi}')$ iff for all $u \neq s$, $v \in S_2, \tilde{\mu}'(u, v) = 0$, the distribution $\mu' : t' \mapsto \tilde{\mu}'(s, t')$ is in $Sat(\varphi')$.*
  2. *If $\exists s^* : s \xrightarrow{*} s^* : act(s^*) \dagger a'$ in $\mathcal{N}_1||\mathcal{N}_2$, then $\forall \tilde{\varphi}' \in C(S_1 \times S_2)$ let $V((s, s'), a', \tilde{\varphi}') = \bot$.*

Note that the first two clauses of Def. 7.14 are the same as for Def. 7.11. The crux of the definition of layered composition lies in the last clause. The first part of this clause defines V in the same way as for parallel composition, in case the action in $\mathcal{N}_2$ is independent of all actions enabled in states reachable from the current state in $\mathcal{N}_1$. The second part of this clause ensures that in case an action a' in $\mathcal{N}_2$ depends on one or more actions in $\mathcal{N}_1$, then it cannot be executed before all the actions in $\mathcal{N}_1$ (on which it depends) have taken place. In other words, layered composition restricts the asynchronous interleaving (for the right component), to only those actions of the right component, which are independent to the actions of the left component.

Remark 7.15 *Note that the parallel and layered composition of two APAs with linear constraint functions may lead to an APA whose constraints are polynomial.*

Example 7.16 *The sequential composition of two APAs $\mathcal{N}_1, \mathcal{N}_2$ (Fig. 7.2 (left and middle)) is shown in Fig. 7.2 (right). The parallel composition of $\mathcal{N}_1, \mathcal{N}_2$ is illustrated in Fig. 7.3 (left). If we assume that actions a, d are dependent in $\mathcal{N}_1||\mathcal{N}_2$, then the layered composition $\mathcal{N}_1 \bullet \mathcal{N}_2$ is illustrated in Fig. 7.3 (right). Note that the abstract execution in which d is executed before a has been removed in $\mathcal{N}_1 \bullet \mathcal{N}_2$. To keep the figures simple, we do not show constraint functions obtained after composing the two systems.*

Next, we use the above mentioned composition operators for formulating the communication closed layer (CCL) laws as follows:

Theorem 7.17 (CCL laws) *For APAs $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{M}_1$, and $\mathcal{M}_2$, with $\mathcal{N}_1 \ddagger \mathcal{M}_2$ and $\mathcal{M}_1 \ddagger \mathcal{N}_2$, the following communication closed layer (CCL) equivalences hold:*

1. $\mathcal{N}_1 \bullet \mathcal{M}_2 \equiv \mathcal{N}_1||\mathcal{M}_2$ *(IND)*
2. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_2 \equiv \mathcal{N}_1 \bullet (\mathcal{N}_2||\mathcal{M}_2)$ *(CCL-L)*
3. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_1 \equiv (\mathcal{N}_1||\mathcal{M}_1) \bullet \mathcal{N}_2$ *(CCL-R)*
4. $(\mathcal{N}_1 \bullet \mathcal{N}_2)||(\mathcal{M}_1 \bullet \mathcal{M}_2) \equiv (\mathcal{N}_1||\mathcal{M}_1) \bullet (\mathcal{N}_2||\mathcal{M}_2)$ *(CCL)*

Intuitively, CCL laws allow moving the $\bullet$ operator out of brackets such that the two systems are refinement equivalent. These laws are required for carrying out the structural transformations.

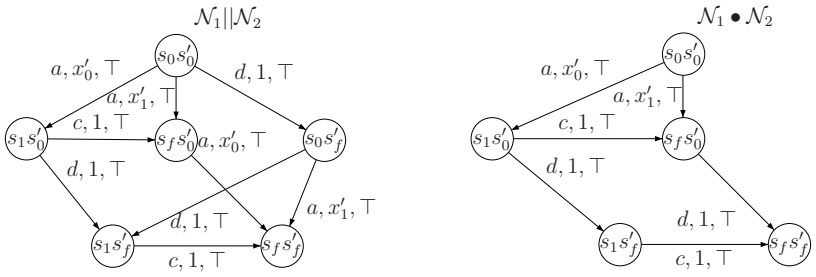

Figure 7.3: Parallel composition $\mathcal{N}_1||\mathcal{N}_2$ (left) and layered composition $\mathcal{N}_1 \bullet \mathcal{N}_2$ (right) where $a \dagger d$

7.3 Partial Order Equivalence and Property Preservation

This section defines the notion of partial order equivalence ($\equiv^*_{po}$) between APAs which is used to prove that if two APAs are po-equivalent then their maximum (resp. minimum) probabilities to reach the set of final states coincide. We start this section by showing that to obtain the maximum (resp. minimum) reachability probabilities for an APA $\mathcal{N}$, it suffices to consider only those implementations that are realisable, i.e., deterministic and satisfy the following condition: every finite path of the implementation is a realisation of some finite abstract execution of $\mathcal{N}$.

Proposition 7.18 *Let $\mathcal{N}$ be an APA. Then we have:*

$$\sup_{\mathcal{P}\in[\![\mathcal{N}]\!]} P^{max}_{\mathcal{P}}(S_f) \;=\; \sup_{\mathcal{Q}\in[\![\mathcal{N}]\!]} P^{max}_{\mathcal{Q}}(S_f)$$

where $\mathcal{Q}$ is realisable.

The above mentioned result can be easily extended to minimum reachability probabilities. From now on, we use $[\![\mathcal{N}]\!]$ to denote the set of implementations of $\mathcal{N}$ that are realisable. For APAs $\mathcal{N}_1$ and $\mathcal{N}_2$, let $\mathcal{N} = \mathcal{N}_1 \bullet \mathcal{N}_2$. Then we define $\mathcal{N}^{\setminus sync}$ as the APA obtained from $\mathcal{N}$ such that it does not have any synchronized transitions (which are present in $\mathcal{N}$ as a result of synchronization over the common actions). Intuitively, this means that abstract executions in $\mathcal{N}$ with synchronized transitions and ending in some final state can be rewritten in $\mathcal{N}^{\setminus sync}$ such that for every abstract execution there are corresponding abstract executions in $\mathcal{N}^{\setminus sync}$ obtained by allowing interleaving on common actions. Note that for every common action, we only allow interleaving (in $\mathcal{N}^{\setminus sync}$) from the state of $\mathcal{N}$ where

the synchronization over that action takes place. For example, let $\mathcal{N}$ have only one transition (a may a-transition) which is a result of synchronization of a must a-transition (from $\mathcal{N}_1$), and a may a-transition (from $\mathcal{N}_2$). In this case $\mathcal{N}^{\setminus sync}$ will have a corresponding[1] sequence of transitions, i.e., a must a-transition (from $\mathcal{N}_1$) followed by a may a-transition (from $\mathcal{N}_2$). This transformation is required as we want to establish the result that layered composition which incorporates synchronization is po-equivalent to sequential composition which does not incorporate synchronization. This means that for any $\mathcal{N} = \mathcal{N}_1; \mathcal{N}_2$, it holds $\mathcal{N}^{\setminus sync} = \mathcal{N}$.

Theorem 7.19 *For APAs $\mathcal{N}_1$ and $\mathcal{N}_2$, let $\mathcal{N} = \mathcal{N}_1 \bullet \mathcal{N}_2$. Then $\forall \mathcal{P} \in [\![\mathcal{N}]\!] \forall \mathcal{D} \in Adv(\mathcal{P})\ \forall \pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P}) : \exists \mathcal{P}' \in [\![\mathcal{N}^{\setminus sync}]\!] \exists \mathcal{D}' \in Adv(\mathcal{P}') \exists \pi' \in Paths^{\mathcal{D}'}_{fin}(\mathcal{P}') : P^{\mathcal{D}}(\pi) = P^{\mathcal{D}'}(\pi')$.*

In stated words, this theorem says that reasoning about $\mathcal{N}^{\setminus sync}$ instead of $\mathcal{N}$ is not a restriction, as the probabilistic behaviour of $\mathcal{N}$ (w.r.t. its implementations) is completely mimicked by $\mathcal{N}^{\setminus sync}$.

Definition 7.20 (po-equivalence) *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two APAs with transition functions V_1 and V_2 respectively. Let $\rho_1 \in Exec_{fin}(\mathcal{N}_1^{\setminus sync})$ and $\rho_2 \in Exec_{fin}(\mathcal{N}_2^{\setminus sync})$. Then $\rho_1 \equiv_{po} \rho_2$ iff there exist finite abstract executions ρ', ρ'' and $\exists c_1, d_1$ with $c_1 \neq d_1$ such that the following holds:*

- $\rho_1 = \rho' c_1 \varphi_1 s d_1 \varphi_2 \rho'' \wedge \rho_2 = \rho' d_1 \varphi_2 s' c_1 \varphi_1 \rho''$, *where* $c_1 \ddagger d_1$.
- $V_1(last(\rho'), c_1, \varphi_1) = V_2(s', c_1, \varphi_1) \wedge V_1(s, d_1, \varphi_2) = V_2(last(\rho'), d_1, \varphi_2)$.
- $\forall i < |\rho'|$: $V_1(s_i, a_{i+1}, \varphi_{i+1}) = V_2(s_i, a_{i+1}, \varphi_{i+1})$.
- $\forall (|\rho'|+2) \leq i < (|\rho'|+|\rho''|+2) : V_1(s_i, a_{i+1}, \varphi_{i+1}) = V_2(s_i, a_{i+1}, \varphi_{i+1})$.

Let $\equiv^*_{po}$, called po-equivalence, *denote the reflexive, transitive closure of $\equiv_{po}$.*

Stated in words, if two finite abstract executions ρ_1, ρ_2 are po-equivalent, then ρ_1 can be obtained from ρ_2 by repeated permutation of adjacent independent actions. Note that the last two conditions of Def. 7.20 are required to ensure that if for example $\rho' = s_0 c_1 \varphi_1 s_1 d_1 \varphi_2 s_2$ where c_1 is a must transition and d_1 is a may transition in $\mathcal{N}_1^{\setminus sync}$, then c_1, d_1 are also must and may transitions in $\mathcal{N}_2^{\setminus sync}$. Let $\mathcal{N}_1 \circ \mathcal{N}_2$ denote $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}$.

[1] In fact, if we would not restrict ourselves to deterministic APAs, then two corresponding transition sequences making a diamond shape would be obtained in $\mathcal{N}^{\setminus sync}$.

Definition 7.21 (LNF) *Let S_f be the set of final states in $\mathcal{N}_1 \circ \mathcal{N}_2$. Then $\rho \in Exec^{S_f}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2)$ is in* layered normal form *(LNF) iff it involves the consecutive execution of actions of $\mathcal{N}_1$, followed by the consecutive execution of actions of $\mathcal{N}_2$.*

Let $Exec^{LNF}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2)$ be the set of all finite abstract executions in $Exec^{S_f}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2)$ that are in LNF.

Next we show that for each finite abstract execution of $Exec^{S_f}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2)$, a po-equivalent abstract exceution in LNF does exist.

Lemma 7.22 (LNF existence) *Let $\mathcal{N}_1, \mathcal{N}_2$ be two APAs. Then we have $\forall \rho \in Exec^{S_f}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2) \exists \rho' \in Exec^{LNF}_{fin}(\mathcal{N}_1 \circ \mathcal{N}_2)$ such that $\rho \equiv^*_{po} \rho'$.*

Next, we define the partial order equivalence between two APAs.

Definition 7.23 (po-equivalence for APAs) *Two APAs $\mathcal{N}_1, \mathcal{N}_2$ are said to be po-equivalent denoted, $\mathcal{N}_1 \equiv^*_{po} \mathcal{N}_2$, iff for $i \in \{1,2\}$: $\forall \rho_i \in Exec^{S_{fi}}_{fin}(\mathcal{N}_i^{\setminus sync}) \exists \rho_{3-i} \in Exec^{S_{f3-i}}_{fin}(\mathcal{N}_{3-i}^{\setminus sync})$ such that $\rho_i \equiv^*_{po} \rho_{3-i}$.*

Theorem 7.24 *For any two APAs $\mathcal{N}_1, \mathcal{N}_2$, it holds: $\mathcal{N}_1 \bullet \mathcal{N}_2 \equiv^*_{po} \mathcal{N}_1; \mathcal{N}_2$.*

In simple words, this theorem says that for each finite execution in layered composition of two APAs, a po-equivalent execution in the their sequential composition does exist and vice versa. This result can be used to replace ; by $\bullet$ provided po-equivalence preserves reachability properties.

Proposition 7.25 *Let $\mathcal{N}$, $\mathcal{N}'$ and $\mathcal{N}_1$ be three APAs such that $\mathcal{N} \equiv^*_{po} \mathcal{N}'$. Then we have:*

$$\mathcal{N}||\mathcal{N}_1 \equiv^*_{po} \mathcal{N}'||\mathcal{N}_1$$

Theorem 7.26 (Property preservation) *Let $\mathcal{N}_1, \mathcal{N}_2$ be two APAs with set of final states S_{f1} and S_{f2} respectively. If $\mathcal{N}_1 \equiv^*_{po} \mathcal{N}_2$ then we have:*

$$P^{max}_{\mathcal{N}_1}(S_{f1}) = P^{max}_{\mathcal{N}_2}(S_{f2})$$

$$P^{min}_{\mathcal{N}_1}(S_{f1}) = P^{min}_{\mathcal{N}_2}(S_{f2})$$

In simple words this theorem states that if two APAs are po-equivalent, then their maximum (resp. minimum) reachability probabilities computed over all the implementations coincide. Results of Theorem 7.24, Proposition 7.25 and

Thm. 7.26 enable us to replace ; with • and vice versa. This replacement along with CCL laws (Thm. 7.17) can be used for state space reduction as follows:

State space reduction. Let $\mathcal{N}_1, \mathcal{N}_2$, $\mathcal{M}_1$ and $\mathcal{M}_2$ be four APAs, and $\mathcal{N} = (\mathcal{N}_1; \mathcal{N}_2)||(\mathcal{M}_1; \mathcal{M}_2)$. Let us say we want to compute $P^{max}_{\mathcal{N}}(S_f)$ or $P^{min}_{\mathcal{N}}(S_f)$. Here, S_f is the set of final states in $\mathcal{N}$. Assume $\mathcal{M}_1 \ddagger \mathcal{N}_2$, $\mathcal{N}_1 \ddagger \mathcal{M}_2$ and $\mathcal{N}_1, \mathcal{N}_2, \mathcal{M}_1, \mathcal{M}_2$ each consist of 20 states. In this case $\mathcal{N}_1; \mathcal{N}_2$ has 39 states which combined with the 39 states of $\mathcal{M}_1; \mathcal{M}_2$ gives 1521 states. We can transform $\mathcal{N}$ in the following way:

$$
\begin{array}{ll}
(\mathcal{N}_1; \mathcal{N}_2)||(\mathcal{M}_1; \mathcal{M}_2) & \\
\equiv^*_{po} & \text{Theorem 7.24, Proposition 7.25} \\
(\mathcal{N}_1 \bullet \mathcal{N}_2)||(\mathcal{M}_1 \bullet \mathcal{M}_2) & \\
\equiv & \text{CCL} \\
(\mathcal{N}_1||\mathcal{M}_1) \bullet (\mathcal{N}_2||\mathcal{M}_2) & \\
\equiv^*_{po} & \text{Theorem 7.24} \\
(\mathcal{N}_1||\mathcal{M}_1); (\mathcal{N}_2||\mathcal{M}_2) &
\end{array}
$$

Note that the transformed system, i.e., $(\mathcal{N}_1||\mathcal{M}_1); (\mathcal{N}_2||\mathcal{M}_2)$ has 799 states.

7.4 Possible Extensions

In this section we briefly discuss the extension of our results to an APA equipped with data variables and rewards.

APA with data. An APA $\mathcal{N}$ can be extended with data variables such that whenever an action is executed its associated data variables are updated according to an arithmetic expression. These data variables can take values in some finite range D. The definitions of satisfaction, and refinement can be slightly modified by imposing an extra condition that ensures that related states have the same valuations. For an APA with data, two actions are said to be dependent if one of the two writes a variable that is read or written by the other action. More formally, two actions a and b are dependent, denoted $a \dagger b$, iff $Write(b) \cap Read(a) \neq \varnothing$ or $Write(a) \cap Read(b) \neq \varnothing$ or $Write(a) \cap Write(b) \neq \varnothing$. Here, $Write(a)$ denotes the set of data variables written by the action a. Similarly, $Read(a)$ denotes the set of data variables read by the action a. Using this dependency relation, our theory can be applied to APAs with data. We do not go into details on these matters here, however, refer the interested reader to [13, 82, 149].

APA with rewards. Rewards are useful for computing, for instance, the resource consumption in a PA under an adversary $\mathcal{D}$. For example in a communication system where a sender and a receiver can transfer messages via an unreliable channel, in this case an interesting measure of interest is the maximum (resp. minimum) expected number of attempts to send a message until correct delivery. An APA $\mathcal{N}$ can be extended with rewards by augmenting state and action pairs, i.e., (s, a) with rewards, which are non-negative real valued numbers. This intuitively means that in every implementation of $\mathcal{N}$, the state and action pairs corresponding to some state and action pair in $\mathcal{N}$ also have the same reward associated with them. The reward associated with a state and action pair, i.e., (s, a) is earned only when the state s is left by executing an action a. In this setting, the expected reward earned along a finite path π of PA $\mathcal{P}$ (under some adversary $\mathcal{D}$) before reaching the final state is obtained by taking the product of $P^{\mathcal{D}}(\pi)$ and the total reward earned along π. The preservation results presented in this paper can be easily extended to APA with rewards, i.e., if two APAs are po-equivalent, then their maximum (resp. minimum) expected rewards earned before reaching the set of final states computed over all the implementations coincide.

7.5 Related Work

APA. Abstract Probabilistic Automata (APAs) were originally defined in [44, 45]. In [44], a complete abstraction and specification theory for PAs was proposed. This theory was later extended to support specifications over dissimilar alphabets, some additional operators, and an APA-embedding of Interface Automata [47]. A tool implementing this theory was reported in [46]. Recently, compositional abstraction techniques for APAs have been proposed, which are based on the notion of common combined transitions [147]. The theory presented in this paper is based on the APA model introduced in [44].

Layering. In the probabilistic context, layering has been applied to the consensus problem to prove complexity lower bounds [114]. The layered composition operator and probabilistic counterparts of the CCL laws have been defined for the PA model [149]. As mentioned before, the feasibility of this approach has been demonstrated on a randomized mutual exclusion algorithm. Most recently, the theory of layering has been extended to modal transition systems (MTSs) [145]. In [145], it has been shown that layering can be used for the state-space reduction of distributed systems that are modeled as a network of acyclic MTSs.

Our results can be viewed as an extension of the layering for PAs [149] and MTSs [145] to APAs.

7.6 Conclusion

This chapter presented a state-space reduction technique for a network of acyclic APAs, based on the notion of layering. We proposed a layered composition operator, and formulated communication closed layer (CCL) laws. Next, we defined the partial order (po) equivalence between APAs, and proved that if two APAs are po-equivalent, then their probabilities to reach the set of final states computed over all the implementations coincide. Finally, we discussed the possible extensions of our results.

7. LAYERED REDUCTION FOR ABSTRACT PROBABILISTIC AUTOMATA

Chapter 8

Interactive Markov Chains

This chapter proposes two equivalence relations for closed IMCs that can be used to reduce the state space before analysis. More specifically, we define interactive Markovian equivalence (IME) that can be seen as a combination of KME (for KSs) and WL (for CTMCs). In other words, IME coincides with KME for any IMC without Markovian states. Similarly, IME coincides with WL for any IMC without interactive states. We show that state space reduction under IME can potentially be larger than for bisimulation for IMCs. In the weaker setting, we define weak interactive Markovian equivalence (WIME) that abstracts from internal or non-observable steps. We show that weak bisimulation for IMCs is strictly finer than WIME. Note that the definitions of bisimulation and weak bisimulation for IMCs (in our case) also take into account state labels. We also show that it is not possible to extend the framework of layering to a network of IMCs. The main contributions of this chapter are as follows:

- We provide a structural definition of IME on closed IMCs, define the quotient under IME and show that bisimulation is strictly finer than IME.
- In the weak setting, we provide a structural definition of WIME on IMCs, define the quotient under WIME and show that weak bisimulation is strictly finer than WIME.
- We show that it is not possible to relate the sequential composition operator and layered (parallel) composition operator by using a partial order relation or simulation relation such that properties of interest are still preserved.

Organisation of this chapter. Section 8.1 defines interactive Markovian equivalence and investigates its relationship with bisimulation. Sections 8.2 de-

fines weak interactive Markovian equivalence and investigates its relationship with weak bisimulation. Section 8.3 shows that the framework of layered reduction cannot be extended to a network of IMCs. Section 8.4 discusses related work. Finally, Section 8.5 concludes the chapter.

8.1 Interactive Markovian Equivalence

Before defining interactive Markovian equivalence, we first define some auxiliary concepts. All the definitions in this section are relative to a closed IMC $\mathcal{I} = (S, s_0, Act, AP, \rightarrow, \Rightarrow, L)$, where $Act = \{\tau\}$. For any state $s \in S$ and $Act = \{\tau\}$, the set of τ-predecessors of s is defined by: $Pred(s, \tau) = \{s' \in S | s' \xrightarrow{\tau} s\}$ and $Pred(s) = \{s' \in S | R(s', s) > 0\} \cup Pred(s, \tau)$. Let for $C \subseteq S$, $Pred(C) = \bigcup_{s \in C} Pred(s)$. Similarly, the set of τ-successors of any state s is defined by: $Post(s, \tau) = \{s' \in S | s \xrightarrow{\tau} s'\}$ and $Post(s) = \{s' \in S | R(s, s') > 0\} \cup Post(s, \tau)$. Let $Post(C) = \bigcup_{s \in C} Post(s)$ and $Post(s, \tau, C) = \{s' \in C | s \xrightarrow{\tau} s'\}$. Throughout this section and Section 8.2 we assume that every state of closed IMC $\mathcal{I}$ has at least one predecessor and the initial state s_0 of $\mathcal{I}$ is distinguished from all other states by a unique label, say \$ (as in the case of KSs, DTMCs and CTMCs (Chapter 2)).

Definition 8.1 *Let $C \subseteq S$, then C is said to be* interactive *closed iff $C \subseteq IS \wedge Pred(C) \subseteq IS$.*

Definition 8.2 *Let $C \subseteq S$, then C is said to be* Markovian *closed iff $C \subseteq MS \wedge Pred(C) \subseteq MS$.*

Let $I(S)$ denote the set of all possible subsets of S that are interactive closed. Let $M(S)$ denote the set of all possible subsets of S that are Markovian closed.

Definition 8.3 (Predecessor based reachability) *For $s \in S$ and $C, D \subseteq S$, the function $Pbr : S \times 2^S \times 2^S \rightarrow \{0, 1\}$ is defined as:*

$$Pbr(s, C, D) = \begin{cases} 1 & \text{if } \exists s' \in Post(s, \tau, C) \text{ s.t. } Post(s', \tau, D) \neq \varnothing \\ 0 & \text{otherwise.} \end{cases}$$

Definition 8.4 (IME) *Equivalence $\mathcal{R}$ on S is an* interactive Markovian equivalence *(IME) if we have:*

1. *$\forall (s_1, s_2) \in \mathcal{R}$ it holds: $L(s_1) = L(s_2)$ and $E(s_1) = E(s_2)$,*

2. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \in I(S)$, $\forall D \in S/_{\mathcal{R}}$ *and* $\forall s', s'' \in Pred(C)$ *it holds:* $Pbr(s', C, D) = Pbr(s'', C, D)$.

3. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \in M(S)$, $\forall D \in S/_{\mathcal{R}}$ *and* $\forall s', s'' \in Pred(C)$ *it holds:* $wr(s', C, D) = wr(s'', C, D)$.

4. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \notin I(S) \wedge C \notin M(S)$, $\forall s_1, s_2 \in C$ *the following holds:*

   - $\forall D \in S/_{\mathcal{R}}$: $R(s_1, D) = R(s_2, D)$,
   - $\forall D \in S/_{\mathcal{R}}$: $Post(s_1, \tau, D) \neq \varnothing \Leftrightarrow Post(s_2, \tau, D) \neq \varnothing$.

States s_1, s_2 *are* IM *related, denoted by* $s_1 \bigtriangledown s_2$, *if* $(s_1, s_2) \in \mathcal{R}$ *for some IME* $\mathcal{R}$.

The first condition asserts that s_1 and s_2 are equally labeled and have identical exit rates. The second condition asserts that for any interactive closed equivalence class C, the predecessor based reachability of going from any two predecessors of C to D via any state in C must be equal. Similarly, third condition requires that for any Markovian closed equivalence class C, the weighted rate (Def. 5.1) of going from any two predecessors of C to D via any state in C must be equal. The last condition requires that for any other case all the states in C should exhibit identical stepwise behavior.

Example 8.5 *For the closed IMC in Fig. 8.1 (left), the equivalence relation induced by the partitioning* $\{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4\}, \{s_5, s_6, s_7\}, \{s_8\}, \{s_9\}\}$ *is an IME relation.*

8.1.1 Quotient IMC

Definition 8.6 *For an IME relation* $\mathcal{R}$ *on* $\mathcal{I}$, *the quotient IMC* $\mathcal{I}/_{\mathcal{R}}$ *is defined by* $\mathcal{I}/_{\mathcal{R}} = (S/_{\mathcal{R}}, s_0', Act, AP, \rightarrow', \Rightarrow', L')$ *where:*

- $S/_{\mathcal{R}}$ *is the set of all equivalence classes under* $\mathcal{R}$,
- $s_0' = C$ *where* $s_0 \in C = [s_0]_{\mathcal{R}}$,
- $\rightarrow' \subseteq S/_{\mathcal{R}} \times Act \times S/_{\mathcal{R}}$ *is defined as follows:* $\frac{C \in I(S) \wedge Pbr(s', C, D) = 1,\ s' \in Pred(C)}{C \xrightarrow{\tau} D}$ *and* $\frac{C \notin I(S) \wedge \exists s \in C, s' \in D : s \xrightarrow{\tau} s'}{C \xrightarrow{\tau} D}$,
- $\Rightarrow' \subseteq S/_{\mathcal{R}} \times \mathbb{R}_{\geq 0} \times S/_{\mathcal{R}}$ *is defined as follows:* $\frac{C \in M(S) \wedge \lambda = wr(s', C, D),\ s' \in Pred(C)}{C \xrightarrow{\lambda} D}$ *and* $\frac{C \notin M(S) \wedge \lambda = R(s, D),\ s \in C}{C \xrightarrow{\lambda} D}$,

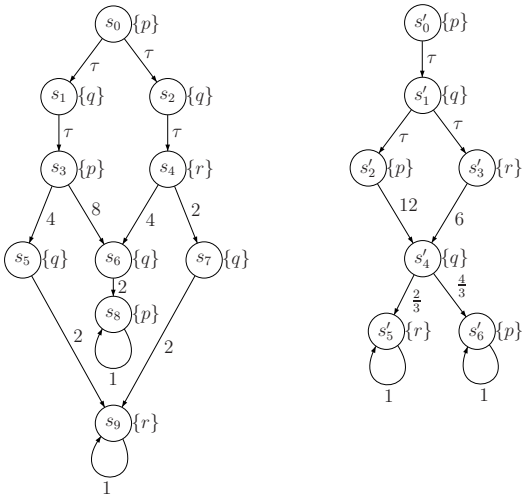

Figure 8.1: An IMC $\mathcal{I}$ (left) and its quotient under an IME $\mathcal{R}$ (right)

- $L'(C) = L(s)$, *where* $s \in C$.

Example 8.7 *The quotient IMC for the Fig. 8.1 (left) under the IME relation with partition* $\{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4\}, \{s_5, s_6, s_7\}, \{s_8\}, \{s_9\}\}$ *is shown in Fig. 8.1 (right).*

Next, we show that any closed IMC $\mathcal{I}$ and its quotient under IME relation are $\bigtriangledown$-related.

Definition 8.8 *Any IMC* $\mathcal{I}$ *and its quotient* $\mathcal{I}/_{\mathcal{R}}$ *under IME* $\mathcal{R}$ *are* $\bigtriangledown$*-related, denoted by* $\mathcal{I} \bigtriangledown \mathcal{I}/_{\mathcal{R}}$*, if and only if there exists an IME relation* $\mathcal{R}^*$ *defined on the disjoint union of state space* $S \uplus S/_{\mathcal{R}}$ *such that*

$$\forall C \in S/_{\mathcal{R}}, \forall s \in C \implies (s, C) \in \mathcal{R}^*$$

.

Theorem 8.9 *Let* $\mathcal{I}$ *be a closed IMC and* $\mathcal{R}$ *be an IME on* $\mathcal{I}$*. Then* $\mathcal{I} \bigtriangledown \mathcal{I}/_{\mathcal{R}}$.

Remark 8.10 *We know that for any closed IMC where* $MS = \varnothing$*, the definition of IME coincides with that of KME. We also know that union of KMEs is not necessarily a KME (Example 3.10), and therefore it is easy to check that union of IMEs is not necessarily an IME. Similarly, we can show that IMEs can be used for repeated minimization of a closed IMC (para. 3.1.1).*

8.1.2 IME vs. Bisimulation

Definition 8.11 *(Strong bisimulation) Let $\mathcal{I} = (S, s_0, Act, AP, \rightarrow, \Rightarrow, L)$ be a closed IMC. An equivalence relation $\mathcal{R} \subseteq S \times S$ is a strong bisimulation on $\mathcal{I}$ if for any $(s_1, s_2) \in \mathcal{R}$ and equivalence class $C \in S/_{\mathcal{R}}$ the following holds:*

- $L(s_1) = L(s_2)$,
- $R(s_1, C) = R(s_2, C)$,
- $Post(s_1, \tau, C) \neq \varnothing \Leftrightarrow Post(s_2, \tau, C) \neq \varnothing$.

States s_1 and s_2 are strongly bisimilar, denoted $s_1 \sim s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some strong bisimulation $\mathcal{R}$.

Theorem 8.12 *$\sim$ is strictly finer than IME.*

This theorem says that state space reduction under IME can potentially be larger than for strong bisimulation.

8.2 Weak Interactive Markovian Equivalence

In this section we define weak interactive Markovian equivalence which results in quotient IMCs that may be significantly smaller than the quotient under IME.

Definition 8.13 (Weak predecessor based reachability) *For $s \in S$ and $C, D \subseteq S$, the function $WPbr : S \times 2^S \times 2^S \rightarrow \{0, 1\}$ is defined as:*

$$WPbr(s, C, D) = \begin{cases} 1 & \text{if } \exists s' \in Post(s, \tau, C), s'' \in D \text{ s.t. } s' \xrightarrow{\tau^+} s'' \\ 0 & \text{otherwise.} \end{cases}$$

where $s' \xrightarrow{\tau^+} s''$ denotes an alternationg sequence of states and τ transitions, i.e., $\pi = s' \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \ldots s_n \xrightarrow{\tau} s''$, where $n \geq 0$ and $s_i \in C, i = 1, \ldots, n$.

Remark 8.14 *Note that if $n = 0$ then $s' \xrightarrow{\tau^+} s''$ denotes $s' \xrightarrow{\tau} s''$, i.e., one step reachability.*

Definition 8.15 (WIME) *Equivalence $\mathcal{R}$ on S is a* weak interactive Markovian equivalence *(WIME) if we have:*

1. *$\forall (s_1, s_2) \in \mathcal{R}$ it holds: $L(s_1) = L(s_2)$.*

2. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \in I(S)$, $\forall D \in S/_{\mathcal{R}}$ *s.t.* $C \neq D$ *and* $\forall s', s'' \in Pred(C)$ *s.t.* $s', s'' \notin C$ *it holds:* $WPbr(s', C, D) = WPbr(s'', C, D)$,

3. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \in M(S)$, $\forall D \in S/_{\mathcal{R}}$ *and* $\forall s', s'' \in Pred(C)$ *it holds:* $wr(s', C, D) = wr(s'', C, D)$ *and* $\forall s_1, s_2 \in C : E(s_1) = E(s_2)$.

4. $\forall C \in S/_{\mathcal{R}}$ *s.t.* $C \notin I(S) \wedge C \notin M(S)$, $\forall s_1, s_2 \in C$ *the following holds:*

   - $\forall D \in S/_{\mathcal{R}}$ *s.t.* $C \neq D : \exists s' \in D : s_1 \xrightarrow{\tau^+} s' \Leftrightarrow \exists s'' \in D : s_2 \xrightarrow{\tau^+} s''$,
   - $\forall D \in S/_{\mathcal{R}}$ *s.t.* $C \neq D : s_1 \xrightarrow{\tau^*} s' \wedge s' \in MS \Rightarrow s_2 \xrightarrow{\tau^*} s'' \wedge s'' \in MS \wedge R(s', D) = R(s'', D)$.

   where $s_1 \xrightarrow{\tau^*} s'$ *denotes an alternating sequence of states and* τ *transitions, i.e.,* $\pi = s_{01} \xrightarrow{\tau} s_{02} \xrightarrow{\tau} s_{03} \ldots s_{0n}$, *where* $n \geq 1, s_1 = s_{01}, s_{0n} = s'$ *and* $s_{0i} \in C$, $i = 1, \ldots, n$.

States s_1, s_2 *are* WIM *related, denoted by* $s_1 \veebar s_2$, *if* $(s_1, s_2) \in \mathcal{R}$ *for some WIME* $\mathcal{R}$.

The first condition asserts that s_1 and s_2 are equally labeled. The second condition asserts that for any interactive closed equivalence class C, the weak predecessor based reachability of going from any two predecessors of C (that are not in C) to D (where $C \neq D$) must be equal. Similarly, third condition requires that for any Markovian closed equivalence class C, the weighted rate of going from any two predecessors of C to D via any state in C must be equal and all the states in C have identical exit rates. The last condition requires that for any other case, all the states in C should reach the same equivalence classes in one or more steps and all of them should reach a state in zero or more steps which has the same rate of moving to any D (where $C \neq D$).

Remark 8.16 *Note that if* $n = 1$ *then* $s_1 \xrightarrow{\tau^*} s'$ *denotes* s_1.

Example 8.17 *For the closed IMC in Fig. 8.2 (left), the equivalence relation induced by the partitioning* $\{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4\}, \{s_5, s_6, s_7\}, \{s_8\}, \{s_9\}\}$ *is a WIME relation.*

8.2.1 Quotient IMC

Definition 8.18 *For WIME relation* $\mathcal{R}$ *on* $\mathcal{I}$, *the quotient IMC* $\mathcal{I}/_{\mathcal{R}}$ *is defined by* $\mathcal{I}/_{\mathcal{R}} = (S/_{\mathcal{R}}, s'_0, Act, AP, \rightarrow', \Rightarrow', L')$ *where:*

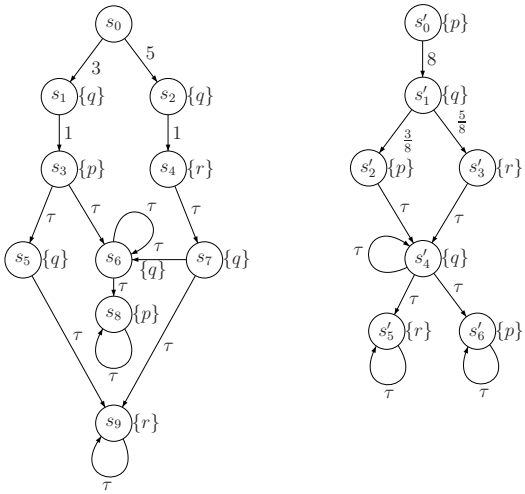

Figure 8.2: An IMC $\mathcal{I}$ (left) and its quotient under a WIME $\mathcal{R}$ (right)

- $S/_{\mathcal{R}}$ *is the set of all equivalence classes under* $\mathcal{R}$,
- $s'_0 = C$ *where* $s_0 \in C = [s_0]_{\mathcal{R}}$,
- $\rightarrow' \subseteq S/_{\mathcal{R}} \times Act \times S/_{\mathcal{R}}$ *is defined as follows:*
  $\frac{C \in I(S) \wedge WPbr(s',C,D) = 1 \wedge C \neq D,\ s' \in Pred(C), s' \notin C}{C \xrightarrow{\tau} D}$ *and* $\frac{C \in I(S) \wedge \exists s \in C : s \xrightarrow{\tau^+} s}{C \xrightarrow{\tau} C}$
  and $\frac{C \notin I(S) \wedge \exists s \in C, s' \in D : s \xrightarrow{\tau} s'}{C \xrightarrow{\tau} D}$,
- $\Rightarrow' \subseteq S/_{\mathcal{R}} \times \mathbb{R}_{\geq 0} \times S/_{\mathcal{R}}$ *is defined as follows:*
  $\frac{C \in M(S) \wedge \lambda = wr(s',C,D),\ s' \in Pred(C)}{C \xrightarrow{\lambda} D}$ *and* $\frac{C \notin M(S) \wedge \lambda = R(s,D),\ s \in C}{C \xrightarrow{\lambda} D}$,
- $L'(C) = L(s)$, *where* $s \in C$.

Example 8.19 *The quotient IMC for the Fig. 8.2 (left) under the WIME relation with partition* $\{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4\}, \{s_5, s_6, s_7\}, \{s_8\}, \{s_9\}\}$ *is shown in Fig. 8.2 (right).*

Definition 8.20 *Any IMC* $\mathcal{I}$ *and its quotient* $\mathcal{I}/_{\mathcal{R}}$ *under WIME* $\mathcal{R}$ *are* $\trianglelefteq$*-related, denoted by* $\mathcal{I} \trianglelefteq \mathcal{I}/_{\mathcal{R}}$, *if and only if there exists a WIME relation* $\mathcal{R}^*$ *defined on the disjoint union of state space* $S \uplus S/_{\mathcal{R}}$ *such that*

$$\forall C \in S/_{\mathcal{R}}, \forall s \in C \implies (s, C) \in \mathcal{R}^*$$

.

Theorem 8.21 *Let $\mathcal{I}$ be a IMC and $\mathcal{R}$ be a WIME on $\mathcal{I}$. Then $\mathcal{I} \trianglerighteq \mathcal{I}/_{\mathcal{R}}$.*

Remark 8.22 *It is easy to check that WIMEs can be used for repeated minimization of a closed IMC and union of WIMEs is not necessarily a WIME.*

8.2.2 WIME vs. Weak Bisimulation

Definition 8.23 *Let $\mathcal{I} = (S, s_0, Act, AP, \rightarrow, \Rightarrow, L)$ be a closed IMC. An equivalence relation $\mathcal{R} \subseteq S \times S$ is a weak bisimulation on $\mathcal{I}$ if for any $(s_1, s_2) \in \mathcal{R}$ and equivalence class $C \in S/_{\mathcal{R}}$ s.t. $C \neq [s_1]_{\mathcal{R}}$ the following holds:*

- $L(s_1) = L(s_2)$,
- $\exists s' \in C : s_1 \xrightarrow{\tau^+} s' \Leftrightarrow \exists s'' \in C : s_2 \xrightarrow{\tau^+} s''$,
- $s_1 \xrightarrow{\tau^*} s' \wedge s' \in MS \Rightarrow s_2 \xrightarrow{\tau^*} s'' \wedge s'' \in MS \wedge R(s', C) = R(s'', C)$ *for some* $s'' \in S$.

where $s_1 \xrightarrow{\tau^*} s'$ denotes an alternating sequence of states and τ transitions, i.e., $\pi = s_{01} \xrightarrow{\tau} s_{02} \xrightarrow{\tau} s_{03} \ldots s_{0n}$, where $n \geq 1, s_1 = s_{01}, s_{0n} = s'$ and $(s_1, s_{0i}) \in \mathcal{R}, i = 1, \ldots, n$. States s_1 and s_2 are weakly bisimilar, denoted $s_1 \approx s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$.

Remark 8.24 *Note that different notations have been used for defining $s_1 \xrightarrow{\tau^*} s'$ in Def. 8.15 and Def. 8.23, but both of them convey the same meaning. The reason behind using different notations is that weak bisimulation [74, 75] checks conditions for any $(s_1, s_2) \in \mathcal{R}$, whereas we check it for any $s_1, s_2 \in C$, where $C \in S/_{\mathcal{R}}$.*

Theorem 8.25 *$\approx$ is strictly finer than WIME.*

This theorem asserts that WIME can achieve a larger state space reduction as compared to weak bisimulation.

Theorem 8.26 *IME is strictly finer than WIME.*

This theorem asserts that WIME can achieve a larger state space reduction as compared to IME.

8.3 Layering for Interactive Markov Chains - A Failed Attempt

IMCs can be used for the modeling and analysis of distributed algorithms with random times, e.g., mutual exclusion algorithm with random times [57]. In [115], the modeling and analysis of mutual exclusion algorithm with random times has been carried out using IMCs. In this setting, read and write operations performed on data variables and registers are assumed to take a randomly distributed amount of time to finish which is governed by negative exponential distributions. These variables and registers are required to make sure that the critical section can be accessed in a mutually exclusive manner. We do not go into details on these matters here, however, refer interested reader to [115].

Our focus in this section is to show that it is not possible to extend the framework of layering to a network of IMCs. In other words, while analysing a distributed algorithm modeled as a network of IMCs, it is not possible to reduce the state space by performing layered structural transformations. More specifically, we show that it is not possible to relate the sequential composition operator and layered (parallel) composition operator by using a partial order relation or simulation relation such that properties of interest are still preserved. Note that this relation is very important for carrying out the structural transformations within the framework of CCL laws (as shown in Section. 7.3). For the sake of simplicity we assume that the states of IMC models are not labeled. We also assume that IMC models are acyclic and have a single final state ($S_f = \{s_f\}$) s.t. all the states (except the final state) have at least one outgoing transition (these assumptions are similar to Chapter 6 and Chapter 7). Thus an IMC is a tuple $\mathcal{I} = (S, s_0, Act, \rightarrow, \Rightarrow, S_f)$ where $S, s_0, Act, \rightarrow, \Rightarrow$ are defined as for Def. 2.28 and S_f is the set of final states with $|S_f| = 1$. Next, we define the composition operators for two IMCs. We propose a sequential composition operator and recall the parallel composition from [75].

Definition 8.27 (Sequential composition) *Given IMCs* $\mathcal{I}_i = (S, s_{0i}, Act_i, \rightarrow_i, \Rightarrow_i, \{s_{fi}\})$, *where* $i \in \{1, 2\}$ *with* $S_1 \cap S_2 = \varnothing$. *Their sequential composition, denoted* $\mathcal{I}_1; \mathcal{I}_2$, *is the IMC* $(S, s_{01}, Act_1 \cup Act_2, \rightarrow, \Rightarrow, \{s_{f2}\})$, *where* $S = S_1 \setminus \{s_{f1}\} \cup S_2$, $\rightarrow = \rightarrow' \cup \rightarrow_2$ *and* $\Rightarrow = \Rightarrow' \cup \Rightarrow_2$. *Here* $\rightarrow' = \rightarrow_1 [s_{02} \leftarrow s_{f1}]$ *is defined by*

$$(s, a, s') \in \rightarrow' \text{ if } (s, a, s') \in \rightarrow_1 \wedge s' \neq s_{f1}, \text{ and } (s, a, s_{02}) \in \rightarrow' \text{ otherwise.}$$

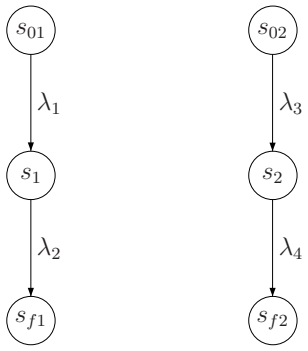

Figure 8.3: IMCs $\mathcal{I}_1$ and $\mathcal{I}_2$

Similarly, $\Rightarrow' = \Rightarrow_1 [s_{02} \leftarrow s_{f1}]$ *is defined by*

$$(s, \lambda, s') \in \Rightarrow' \text{ if } (s, \lambda, s') \in \Rightarrow_1 \wedge s' \neq s_{f1}, \text{ and } (s, \lambda, s_{02}) \in \Rightarrow' \text{ otherwise.}$$

Intuitively, sequential composition of two IMCs $\mathcal{I}_1$ and $\mathcal{I}_2$ requires executing the actions/delay transitions of $\mathcal{I}_1$ followed by actions/delay transitions of $\mathcal{I}_2$. Note that all the incoming transitions to state s_{f1} are redirected to s_{02}. Here, s_{01}, s_{f2} are the new initial and final states in the resulting IMC, respectively.

Definition 8.28 (Parallel composition [75]) *Given IMCs* $\mathcal{I}_i = (S, s_{0i}, Act_i, \rightarrow_i, \Rightarrow_i, \{s_{fi}\})$*, where* $i \in \{1, 2\}$ *with* $S_1 \cap S_2 = \varnothing$*. The parallel composition of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$*, denoted* $\mathcal{I}_1 || \mathcal{I}_2$*, is the IMC* $(S_1 \times S_2, (s_{01}, s_{02}), Act_1 \cup Act_2, \rightarrow, \Rightarrow, \{(s_{f1}, s_{f2})\})$ *where* $\rightarrow$ *and* $\Rightarrow$ *are defined as follows:*

- $s_1 \xrightarrow{a}_1 s_1'$ *and* $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \in Act_1 \cap Act_2$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2')$
- $s_1 \xrightarrow{a}_1 s_1'$ *and* $a \notin Act_1 \cap Act_2$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1', s_2)$ *for any* $s_2 \in S_2$
- $s_2 \xrightarrow{a}_2 s_2'$ *and* $a \notin Act_1 \cap Act_2$ *implies* $(s_1, s_2) \xrightarrow{a} (s_1, s_2')$ *for any* $s_1 \in S_1$
- $s_1 \overset{\lambda}{\Rightarrow}_1 s_1'$ *implies* $(s_1, s_2) \xrightarrow{\lambda} (s_1', s_2)$ *for any* $s_2 \in S_2$
- $s_2 \overset{\lambda}{\Rightarrow}_2 s_2'$ *implies* $(s_1, s_2) \xrightarrow{\lambda} (s_1, s_2')$ *for any* $s_1 \in S_1$

Parallel composition allows synchronization on common actions and interleaving on disjoint actions. The last two conditions assert that IMCs can delay independently.

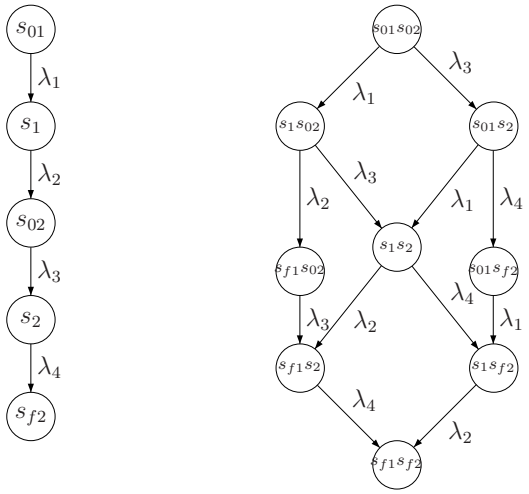

Figure 8.4: Sequential composition $\mathcal{I}_1;\mathcal{I}_2$ (left) and parallel composition $\mathcal{I}_1||\mathcal{I}_2$ (right)

Example 8.29 *The sequential composition of two IMCs $\mathcal{I}_1, \mathcal{I}_2$ (Fig. 8.3) is shown in Fig. 8.4 (left). The parallel composition of $\mathcal{I}_1, \mathcal{I}_2$ is illustrated in Fig. 8.4 (right).*

From the previous chapters (Chapter 6, and Chapter 7), we know that for developing the theory of layering, a partial order equivalence relation needs to be established between the sequential and layered/parallel composition operators. Next, we show that it is not possible to establish a partial order equivalence between IMCs s.t. reachability probabilities are preserved.

Example 8.30 *Consider the IMCs shown in Fig. 8.4. It can be checked that the maximal (resp. minimal) probability to reach the set of final states from initial state within t time units is different for the IMCs. In other words, sequential composition (left) and parallel composition (right) do not have the same probability for reachability properties. This is also true if we are considering computing the extremal expected*[1] *time properties for IMCs. This is due to the fact that the exit rate of s_{01} is smaller than the exit rate of (s_{01}, s_{02}). From this observation we can conclude that any definition of po-equivalence that relates ; and || composition operators would not preserve the reachability probabilities.*

[1]We do not go into details on how to compute the extremal reachability and expected time probabilities, however, refer interested reader to [71, 75, 160].

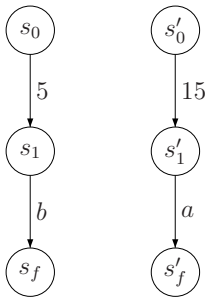

Figure 8.5: IMCs $\mathcal{I}_1$ and $\mathcal{I}_2$

Next, we show that it is not even possible to relate the ; and || composition operators by simulation relation for IMCs. Let $s \overset{\tau}{\not\rightarrow}$ denote a predicate that is true if and only if s has no outgoing τ-transition.

Definition 8.31 (Strong simulation [75]) *For IMC* $\mathcal{I} = (S, s_0, Act, \rightarrow, \Rightarrow, S_f)$, $\mathcal{R} \subseteq S \times S$ *is a simulation relation, iff for any* $(s,t) \in \mathcal{R}$ *it holds:*

- *for any* $a \in Act$ *and* $s' \in S$, $s \xrightarrow{a} s'$ *implies* $t \xrightarrow{a} t'$ *and* $(s',t') \in \mathcal{R}$ *for some* $t' \in S$,
- $s \overset{\tau}{\not\rightarrow}$ *implies* $E(s) \leq E(t)$,
- $s \overset{\tau}{\not\rightarrow}$ *implies for distributions* $\mu = P(s,\cdot)$ *and* $\mu' = P(s',\cdot)$ *there exists* $\Delta : S \times S \rightarrow [0,1]$ *such that for all* $u, u' \in S$*:*
  1. $\Delta(u,u') > 0 \implies (u,u') \in \mathcal{R}$
  2. $\Delta(u,S) = \mu(s)$
  3. $\Delta(S,u') = \mu'(u')$

We write $s \preceq s'$ if $(s,s') \in \mathcal{R}$ for some simulation $\mathcal{R}$ and $\mathcal{I} \preceq \mathcal{I}'$ for IMCs $\mathcal{I}$ and $\mathcal{I}'$ with initial states s_0 and s'_0, if $s_0 \preceq s'_0$ in the disjoint union of $\mathcal{I}$ and $\mathcal{I}'$. In case of two different IMCs $\mathcal{I}$ and $\mathcal{I}'$, we impose an extra condition which requires that finals state in $\mathcal{I}$ can only be related to final state in $\mathcal{I}'$ and vice versa. The last condition of Def. 8.31 requires the existence of a weight function Δ that distributes μ of s to μ' of s' such that only related states obtain a positive weight, and the total probability mass of u that is assigned by Δ coincides with $\mu(u)$ and symmetrically for u'.

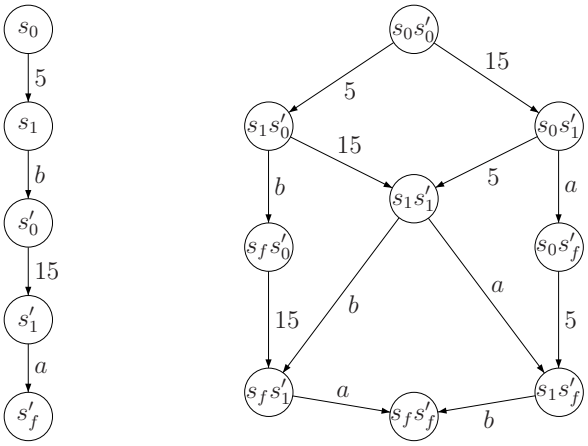

Figure 8.6: Sequential and parallel composition of $\mathcal{I}_1$ and $\mathcal{I}_2$

Example 8.32 *Consider the two IMCs $\mathcal{I}_1$ and $\mathcal{I}_2$ shown in Fig. 8.5. The sequential and parallel composition of $\mathcal{I}_1$ and $\mathcal{I}_2$ is shown in Fig. 8.6. Here $s_0 \not\simeq s_0s'_0$. This is because it is not possible to fulfill the weight condition of Def. 8.31, as the distribution to move to the states s_1 and $s_1s'_0$ is different, i.e., $P(s_0, s_1) = 1 \neq \frac{1}{4} = P((s_0, s'_0), (s_1, s'_0))$. This example shows that parallel composition of two IMCs does not simulate the sequential composition.*

From Example 8.30 and Example 8.32 it can be seen that it is not possible to relate the sequential and parallel composition operators, and therefore the state space of a distributed algorithm modeled as a network of IMCs cannot be reduced using the layered transformation.

8.4 Related Work

Notions of strong and weak bisimulation for IMCs have been studied in [74, 75]. Strong simulation relation for IMCs has been proposed in [75]. Preservation of extremal time-bounded reachability properties by bisimulation and weak bisimulation for IMCs has been proved in [75]. A more aggressive abstraction framework for IMCs that is congruence w.r.t. parallel composition has been defined in [90, 94]. This reduction technique is based on three-valued abstraction and yields lower bounds for minimal and upper bounds for maximal timed reachability properties. In the linear time setting, several variants of trace equivalences

have been analysed in [158] using button pushing experiments. More specifically, various notions of trace equivalence have been studied in [158], based on different types of adversaries (schedulers) that are used to resolve nondeterministic choices. The relationship between IME/WIME and different notions of trace equivalence studied in [158] is not clear.

8.5 Conclusions

This chapter considered two equivalence relations for reducing the state space of closed IMCs. We proposed interactive Markovian equivalence (IME), defined quotient under this relation and proved that bisimulation is strictly finer than IME. In the weak setting, we have proposed weak interactive Markovian equivalence WIME, defined quotient under this relation and proved that weak bisimulation is strictly finer than WIME. We have also shown that it is not possible to extend the framework of layered reduction to a network of IMCs.

Chapter 9

Conclusions and Future Work

We have presented state-space reduction techniques for a range of nondeterministic and probabilistic systems. The reduction techniques for these systems are based on the notions of equivalence relations and layering. We started with the purely nondeterministic setting, i.e., Kripke structures and provide structural definitions of Kripke minimization equivalence (KME) and weak Kripke minimization equivalence (WKME). We have shown that KME and WKME allow for a more aggressive state space reduction than strong bisimulation and divergence-sensitive stutter bisimulation, respectively. Next, we have established the preservation results for linear-time (LT) and stutter-insensitive LT properties under KME and WKME quotienting. We have also proved that KME is compositional w.r.t. synchronous parallel composition for KSs.

For discrete-time probabilistic systems, we have defined weighted probabilistic equivalence (WPE) on discrete-time Markov chains (DTMCs). It has been proved that WPE is strictly coarser than bisimulation for DTMCs, and preserves the probability of satisfying ω-regular properties. As a side result, we have shown that these preservation results can be extended to discrete-time Markov reward models (DMRMs), and WPE is compositional w.r.t. synchronous parallel composition for DTMCs.

In the continuous-time setting, we have provided a structural definition of weighted lumpability (WL) on continuous-time Markov chains (CTMCs). We have shown that WL is strictly coarser than bisimulation for CTMCs, and the probability of satisfying a deterministic timed automaton (DTA) specification is preserved under WL quotienting. This preservation result has also been extended to metric temporal logic (MTL) specifications.

In the second part of this thesis, we have focused on layering based state space

reduction for distributed systems modeled as a network of acyclic modal transition systems (MTSs) and abstract probabilistic automata (APAs). For MTSs, we have defined layered composition operator and proved communication closed layer (CCL) laws. Next, we have defined a partial order (po) equivalence between MTSs and prove that it preserves existential ($\exists$) and universal ($\forall$) reachability properties. Similarly, for APAs we have defined a layered composition operator, formulated CCL laws and established a po equivalence. We have proved the preservation of extremal reachability probabilities under po equivalence. We have also shown that the theory of layering cannot be extended to IMCs, which are useful for modeling distributed algorithms with random times. In addition, we have defined interactive Markovian equivalence (IME) and weak interactive Markovian equivalence (WIME) on closed interactive Markov chains (IMCs). Finally, we have investigated their relationship with bisimulation and weak bisimulation (for closed IMCs), respectively.

Future work

- **Algorithm**: As mentioned before, in [159], a polynomial-time algorithm has been proposed for computing WL relation on CTMCs. This algorithm repeatedly computes the WL quotient system until no further minmization is possible. Initial experiments with two academic case studies have shown that WL can be used for substantial reduction in size of certain CTMC models, e.g., incremental service systems. This algorithm can easily be adapted to compute KME (Chapter 3) on KSs and WPE (Chapter 4) on DTMCs. Unfortunately, the algorithm is currently slow, and therefore an interesting direction of research involves improving the time complexity of this algorithm by changing the way weighted rates are computed. Another challenge is to use more efficient data structures for keeping track of rates to blocks. It needs to be investigated if it is possible to adapt this algorithm to compute weak KME (WKME) on KSs and IME/WIME (Chapter 8) on closed IMCs. Finally, it would be also interesting to investigate other classes of systems where WL/WPE/KS can provide better state space reductions as compared to bisimulation.

- **Property preservation**: For IME and WIME (Chapter 8), a challenging direction of future research would be to explore the linear real-time properties that are preserved by these relations. Some example properties that can be investigated are as follows: extremal timed reachability probabili-

ties [160], extremal expected time [71] and long-run average objectives [71]. For timed reachability, it would be interesting to see if an approach similar to [117] (for CTMDPs) can be used for establishing the preservation results. Recently, in [137] it has been proved that bisimulation for Markov automata preserves extremal expected time and long-run average objectives. This can be used as a starting point for establishing preservation results under IME/WIME quotienting.

- **Layering**: It would be interesting to apply this technique to practical case studies which involve modeling distributed systems using MTSs and APAs. An example case study where layering based structural transformations can be useful for reducing the state space is randomized Byzantine agreement protocol [97]. Another interesting direction of research is to extend the theory of layering for PAs/APAs [144, 149] and timed automata (TAs) [124] to probabilistic timed automata (PTAs) [100].

Appendix A

Appendix

A.1 Proofs of Chapter 3

Proof of Theorem 3.8
Proof: Let $\mathcal{K} = (S, \rightarrow, AP, L, s_0)$ be a Kripke structure and $\mathcal{K}/_{\mathcal{R}} = (S/_{\mathcal{R}}, \rightarrow'$ $, AP, L', s_0')$ be one of its quotient under KME $\mathcal{R}$. Since we have defined the KME relation on a single state space, to prove this theorem we take the disjoint union $S \uplus S/_{\mathcal{R}}$. Let us define a relation $\mathcal{R}' \subseteq (S \uplus S/_{\mathcal{R}}) \times (S \uplus S/_{\mathcal{R}})$ with $\{(s, C) | s \in C\} \subseteq \mathcal{R}'$. Let $\mathcal{R}^*$ be the reflexive, symmetric and transitive of closure of $\mathcal{R}'$. Now we prove that $\mathcal{R}^*$ is a KME relation. This is done by checking both conditions of Def. 3.2. Let $(s, C) \in \mathcal{R}^*$. The proofs for pairs (s, s'), (C, s), and (C, C) are similar and omitted.

1. $L'(C) = L(s)$ by definition of $\mathcal{K}/_{\mathcal{R}}$.

2. Next we prove that $\forall E, F \in (S \uplus S/_{\mathcal{R}})/_{\mathcal{R}^*}$ and $\forall x_0', x_0'' \in Pred(E)$ it holds $Pbr(x_0', E, F) = Pbr(x_0'', E, F)$. Let $x_0', x_0'' \in Pred(E)$. Consider the following three cases based on the successors of x_0', x_0'' such that these successors are in E.

   2.1) The successors of both x_0', x_0'' belong to S. Since we know that $\mathcal{R}$ is a KME, it follows $Pbr(x_0', E, F) = Pbr(x_0'', E, F)$.

   2.2) The successors of both x_0', x_0'' belong to $S/_{\mathcal{R}}$. In this case we know that $Pbr(x_0', E, F) = Pbr(x_0', E_1, F_1)$ where $E_1 \in E \cap S/_{\mathcal{R}}$ and $F_1 \in F \cap S/_{\mathcal{R}}$. We know from definition of $\mathcal{K}/_{\mathcal{R}}$ that $Pbr(x_0', E_1, F_1) = 1$, i.e., $E_1 \rightarrow' F_1$ iff $\forall s', s'' \in Pred(E_1) \cap S$, $Pbr(s', E_1, F_1) = Pbr(s'', E_1, F_1) =$ 1. Similarly $Pbr(x_0'', E, F) = Pbr(x_0'', E_1, F_1) = 1$ iff $\forall s', s'' \in$

$Pred(E_1)$ s.t. $s', s'' \in S$, $Pbr(s', E_1, F_1) = Pbr(s'', E_1, F_1) = 1$. Thus either $Pbr(x'_0, E, F) = Pbr(x''_0, E, F) = 1$ or $Pbr(x'_0, E, F) = Pbr(x''_0, E, F) = 0$.

2.3) The successors of x'_0, x''_0 belong to S and $S/_{\mathcal{R}}$ respectively. We know from the definition of $\mathcal{K}/_{\mathcal{R}}$, $Pbr(x''_0, E_1, F_1) = 1$, i.e., $E_1 \rightarrow' F_1$ iff $Pbr(x'_0, E_1, F_1) = 1$. Therefore we can conclude that $Pbr(x'_0, E, F) = Pbr(x''_0, E, F)$.

■

Proof of Theorem 3.12

Proof: Let $s_1 \sim s_2$. We prove that both conditions for KME are satisfied, i.e., $\sim \Rightarrow KME$.

- $L(s_1) = L(s_2)$, from the definition of $s_1 \sim s_2$.
- Let $S/_{\sim}$ denotes the set of equivalence classes under strong bisimulation relation. Let $C, D \in S/_{\sim}$, $s_1, s_2 \in C$ and $s', s'' \in Pred(C)$. Since $s_1 \sim s_2$ we know that if s_1 can reach D in one step then s_2 can also do so and vice versa. Thus from any predecessor of C we can reach D in two steps and this holds for any $\mathcal{D} \in S/_{\sim}$. Therefore we can conclude: $Pbr(s', C, D) = Pbr(s'', C, D)$.

Next, we show that $KME \not\Rightarrow \sim$. From Fig. 3.2 (left) it can be observed that $s_3 \sim s_5$, but states s_3 and s_4 cannot be merged under $\sim$, as s_3 can move to s_7 but there is no direct successor s of s_4 with $s \sim s_7$ (Note that $L(s_7) \neq L(s_6)$). ■

Proof of Theorem 3.17

Proof: In order to prove this theorem it is sufficient to show that for each path $\pi_1 \in Paths^{\mathcal{K}}(s_0)$ there exists a path $\pi'_1 \in Paths^{\mathcal{K}/\mathcal{R}}(s'_0)$ s.t. $\pi_1 \star \pi'_1$ and vice versa, where $\pi_1 \star \pi'_1$ iff $s_i \star s'_i$ for all i. Since $s_i \star s'_i \implies L(s_i) = L(s'_i)$ (from definition of KME), we know $\pi_1 \star \pi'_1 \Rightarrow \pi_1 \triangle \pi'_1$. Let $\pi_1 = s_0 \rightarrow s_1 \rightarrow s_2 \ldots \in Paths^{\mathcal{K}}(s_0)$ be a path in $\mathcal{K}$ starting in s_0 and assume $s_0 \star s'_0$. Now we successively define a *corresponding* path in $\mathcal{K}/_{\mathcal{R}}$ starting in s'_0 s.t. the transitions $s_i \rightarrow s_{i+1}$ are matched by transitions $s'_i \rightarrow' s'_{i+1}$ where $s_{i+1} \star s'_{i+1}$. This is done by induction on i.

- **Base Case**: $i = 0$. From Def. 3.5 we know that for each transition $s_0 \rightarrow s_1$ in $\mathcal{K}$ there will be a corresponding transition $C \rightarrow' D$ in $\mathcal{K}/_{\mathcal{R}}$ s.t. $s_0 \in C$, $s_1 \in D$. We also know from Theorem 3.8 that $s_0 \star C$ and $s_1 \star D$. Taking $s'_0 = C$ and $s'_1 = D$ we get the path $s'_0 \rightarrow' s'_1$.

- **Induction step**: Assume $i \geq 0$ and that the path $s'_0 \rightarrow' s'_1 \rightarrow' s'_2 \ldots s'_i$ is already constructed with $s_i \star s'_i$ for $j = 0, \ldots, i$. We consider the transition $s_i \rightarrow s_{i+1}$ in π_1. Since $s_i \star s'_i$ for transition $s_i \rightarrow s_{i+1}$ there will be a corresponding transition $C \rightarrow' D$ with $s_{i+1} \star D$ and $s'_i = C$, $s'_{i+1} = D$. Thus we can construct the path $s'_0 \rightarrow' s'_1 \rightarrow' s'_2 \ldots s'_i \rightarrow' s'_{i+1}$.

Similarly for each path $\pi'_1 \in Paths^{\mathcal{K}/\mathcal{R}}(s'_0)$ we can construct a path $\pi_1 \in Paths^{\mathcal{K}}(s_0)$ s.t. $\pi'_1 \star \pi_1$. Hence $\mathcal{K} \models P \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models P$. ■

Proof of Theorem 3.28

Proof: Proof of this theorem is similar to the proof of Theorem 3.8. ■

Proof of Theorem 3.30

Proof: Let $s_1 \star s_2$. We show that s_1, s_2 are also WKME related, i.e., $KME \Rightarrow WKME$.

- $L(s_1) = L(s_2)$ from definition of KME.
- Let $s_1, s_2 \in C$. Since $s_1 \star s_2$, $\forall D \in S/_{\mathcal{R}}$ and $s', s'' \in Pred(C)$: $Pbr(s', C, D) = Pbr(s'', C, D)$. This condition also satisfies condition 2 of WKME (Def. 3.23) as Def. 3.23 only requires that for $s', s'' \notin C$, $C \neq D$, two (or more) step reachability is satisfied.

Next we show that $WKME \not\Rightarrow KME$. Consider Fig. 3.6 (left), here three a states, i.e., s_3, s_4 and s_5 can be merged under WKME but cannot be merged under KME. ■

Proof of Theorem 3.33

Proof: Let $s_1 \cong^{div} s_2$ and $s_1, s_2 \in C$, $C \in S/_{\cong^{div}}$. We prove that s_1, s_2 are also WKME related. From the definition of $\cong^{div}$ we know that if s_1 can reach any equivalence class $D \in S/_{\cong^{div}}$ then s_2 can also reach D in one or more steps via C and vice versa. This also implies that from every predecessor s of C s.t. $s \notin C$ it is possible to reach the same equivalence classes in two or more steps via C (where we only consider equivalence classes that are different from C). We also know that $L(s_1) = L(s_2)$, from definition of $\cong^{div}$, hence s_1, s_2 are WKME related. Next we show that $WKME \not\Rightarrow \cong^{div}$. Consider Fig. 3.6 (left), here three a states, i.e., s_3, s_4 and s_5 cannot be merged under $\cong^{div}$, as s_4 cannot reach a state labeled with $\varnothing$ but s_3 and s_5 can reach s_7. Similarly s_5 cannot reach a b state but s_3 and s_4 can reach s_6. ■

Proof of Theorem 3.37

Proof: In order to prove this theorem it is sufficient to show that for each path $\pi_1 \in Paths^{\mathcal{K}}(s_0)$, there exists a path $\pi'_1 \in Paths^{\mathcal{K}/\mathcal{R}}(s'_0)$ s.t. $\pi_1 \odot \pi'_1$ (i.e., there

exists an infinite sequence of indices $0 = j_0 < j_1 < j_2 < \ldots$ and $0 = k_0 < k_1 < k_2 < \ldots$ with $s_j \odot s'_k$ for all $j_{r-1} \leq j < j_r$ and $k_{r-1} \leq k < k_r$ where $r = 1, 2, \ldots$). Since $s_i \odot s'_i \Rightarrow L(s_i) = L(s'_i)$, we know $\pi_1 \odot \pi'_1 \Rightarrow \pi_1 \triangleq \pi'_1$. Let $\pi_1 = s_0 \to s_1 \ldots \in Paths^{\mathcal{K}}(s_0)$ and assume $s_0 \odot s'_0$. Now we successively define a *corresponding* path $\pi'_1 \in Paths^{\mathcal{K}/\mathcal{R}}(s'_0)$ in $K/_{\mathcal{R}}$ starting in s'_0 s.t. transition $s_i \to s_{i+1}$ is matched by a sequence of transitions $s'_i \to' u_1 \ldots u_n \to' s'_{i+1}$ $(n \geq 0)$ with $s_{i+1} \odot s'_{i+1}$ and $s'_i \odot u_1 \ldots \odot u_n$. This is done by induction on i.

- **Base Case**: $i = 0$. From Def. 3.25, we know that for each transition $s_0 \to s_1$ in $\mathcal{K}$ s.t. $(s_0, s_1) \notin \mathcal{R}$ there will be a corresponding transition $C \to' D$ in $\mathcal{K}/_{\mathcal{R}}$ s.t. $s_0 \in C$, $s_1 \in D$. We also know from Thm. 3.28 that $s_0 \odot C$ and $s_1 \odot D$. Taking $s'_0 = C$ and $s'_1 = D$ we can construct the path $s'_0 \to' u_1 \to' u_2 \ldots u_n \to' s'_1$. If $(s_0, s_1) \in \mathcal{R}$ then we have two subcases.
  - If there exists an index $j > 1$ with $(s_0, s_j) \notin \mathcal{R}$ with j being minimal, i.e. $(s_{j-1}, s_j) \notin \mathcal{R}$ then we have $s_i \odot s_{i+1}$ for all $i = 0 \ldots j-2$. Since $s_{j-1} \to s_j$ and $s'_0 \odot s_{j-1}$, we know there exists a path $s'_0 \to' u_1 \to' u_2 \ldots u_n \to' s'_1$ with $s'_0 \odot u_1 \odot \ldots \odot u_n$ and $s_j \odot s'_1$.
  - If there does not exist an index $j > 1$ with $(s_0, s_j) \notin \mathcal{R}$ this means $\exists s$ s.t. $s \odot s_0$ and $s \xrightarrow{+} s$. From Def. 3.25 and Thm. 3.28 we have $C \to' C$, with $s \in C$. Taking $C = s'_0$ we can construct the path $s'_0 \to' u_1 \to' u_2 \ldots u_n \to' s'_0$, with $s'_0 \odot u_1 \odot \ldots \odot u_n$.
- **Induction Step**: Assume $i \geq 0$ and the path $\pi' = s'_0 \to' u_1 \ldots u_{n_0} \to' s'_1 \to' u_1 \ldots u_{n_1} \to' s'_2 \ldots s'_i$ is already constructed with $s_i \odot s'_i$. Consider the following three cases:
  - If the transition $s_i \to s_{i+1}$ is s.t. $(s_i, s_{i+1}) \notin \mathcal{R}$ then we have the transition $C \to' D$ in $\mathcal{K}/_{\mathcal{R}}$ where $s_i \in C$ and $s_{i+1} \in D$. Taking $C = s'_i$, $D = s'_{i+1}$ we can construct the path $s'_i \to' u_1 \ldots u_{n_i} \to' s'_{i+1}$ with $s_{i+1} \odot s'_{i+1}$ and $s'_i \odot u_1 \ldots u_{n_i}$. Concatenating this path fragment with π' we can construct the corresponding path.
  - Let the transition $s_i \to s_{i+1}$ is s.t. $(s_i, s_{i+1}) \in \mathcal{R}$ and there exists an index $j > i+1$ with $(s_i, s_j) \notin \mathcal{R}$ with j being minimal, i.e. $(s_{j-1}, s_j) \notin \mathcal{R}$. Since $s_{j-1} \to s_j$ and $s'_i \odot s_{j-1}$, we know there exists a path $s'_i \to' u_1 \ldots u_{n_i} \to' s'_{i+1}$ with $s'_i \odot u_1 \odot \ldots \odot u_{n_i}$ and $s_j \odot s'_{i+1}$. Concatenating this path fragment with π' we can construct the corresponding path.

– Let the transition $s_i \rightarrow s_{i+1}$ is s.t. $(s_i, s_{i+1}) \in \mathcal{R}$ and there does not exist an index $j > i+1$ with $(s_i, s_j) \notin \mathcal{R}$ this means $\exists s$ s.t. $s \odot s_i$ and $s \xrightarrow{+} s$. From Def. 3.25 and Thm. 3.28 we know there is a transition $C \rightarrow' C$ in $\mathcal{K}/_{\mathcal{R}}$ with $s_i \in C$. Thus we can construct the path $s'_i \rightarrow' u_1 \rightarrow' u_2 \ldots u_{n_i} \rightarrow' s'_{i+1}$, with $s'_i \odot u_1 \odot \ldots \odot u_n$ and $s'_i = s'_{i+1}$. Concatenating this path fragment with π' we can construct the corresponding path.

Similarly for each path $\pi'_1 \in Paths^{\mathcal{K}/_{\mathcal{R}}}(s'_0)$ we can construct a path $\pi_1 \in Paths^{\mathcal{K}}(s_0)$ s.t. $\pi'_1 \odot \pi_1$. Hence $\mathcal{K} \models P \Leftrightarrow \mathcal{K}/_{\mathcal{R}} \models P$. ■

Proof of Theorem 3.20

Proof: Let $\mathcal{K} = (S, \rightarrow, AP, L, s_0)$ be a Kripke structure and $\mathcal{K}/_{\mathcal{R}} = (S/_{\mathcal{R}}, \rightarrow', AP, L', s'_0)$ be its quotient under KME $\mathcal{R}$. Let $\mathcal{K}_1 = (S_1, \rightarrow_1, AP_1, L_1, s''_0)$ be a Kripke structure composed in parallel to $\mathcal{K}$ and $\mathcal{K}/_{\mathcal{R}}$, i.e., $\mathcal{K} \otimes \mathcal{K}_1$, $\mathcal{K}/_{\mathcal{R}} \otimes \mathcal{K}_1$. Since we have defined KME relation on a single state space, to prove this theorem we take the disjoint union $(S \times S_1) \uplus (S/_{\mathcal{R}} \times S_1)$. Let us define a relation $\mathcal{R}' \subseteq ((S \times S_1) \uplus (S/_{\mathcal{R}} \times S_1)) \times ((S \times S_1) \uplus (S/_{\mathcal{R}} \times S_1))$ such that $\mathcal{R}' = \{((s,t),(C,t)) | s \in C, t \in S_1\}$, $C \in S/_{\mathcal{R}}$. Let $\mathcal{R}^*$ be the the reflexive, symmetric and transitive closure of $\mathcal{R}'$ Now we prove that $\mathcal{R}^*$ is a KME relation. This is done by checking both conditions of Def. 3.2. Let $((s,t),(C,t)) \in \mathcal{R}^*$.

1. From the definition of $\mathcal{K}/_{\mathcal{R}}$ we know that, $L'(C) = L(s)$. Nowe we have to show that: $L(s,t) = L(C,t)$. From Def. 3.19 we know that $L(s,t) = L(s) \cup L(t) = L(C) \cup L(t) = L(C,t)$.

2. Let E, F be the equivalence classes under relation $\mathcal{R}^*$. Next we prove that $\forall E, F \in ((S \times S_1) \uplus (S/_{\mathcal{R}} \times S_1))/_{\mathcal{R}^*}$ and $\forall x'_0, x''_0 \in Pred(E)$ it holds $Pbr(x'_0, E, F) = Pbr(x''_0, E, F)$. Let $x'_0, x''_0 \in Pred(E)$. Consider the following three cases based on the successors of x'_0, x''_0 such that these successors are in E.

   a) The successors of both x'_0, x''_0 belong to $S \times S_1$. Since $C \in S/_{\mathcal{R}}$ we know that $Pbr(s'_0, C, D) = Pbr(s''_0, C, D)$, where $s'_0, s''_0 \in Pred(C)$. Since parallel composition is the synchronous product, where two Kripke structures move in a lock step fashion, i.e., $\forall s_1, s_2 \in C$ and $t \in S_1$, $\frac{s_1 \rightarrow s_3 \wedge s_2 \rightarrow s_4 \wedge t \rightarrow_1 t'}{(s_1,t) \rightarrow_\star (s_3,t') \wedge (s_2,t) \rightarrow_\star (s_4,t')}$, where $\rightarrow_\star$ is the transition relation on $S \times S_1$. That is all the transitions from an equivalence class, say from $C \in S/_{\mathcal{R}}$ to $D \in S/_{\mathcal{R}}$ now are transitions from (C,t) to (D,t') and

therefore the two step reachability from predecessors to (D, t') via (C, t) is still preserved. Thus: $Pbr(x_0', E, F) = Pbr(x_0'', E, F)$.

b) The successors of both x_0', x_0'' belong to $(S/_{\mathcal{R}} \times S_1)$. We have $Pbr(x_0', E, F) = Pbr(x_0', E_1, F_1)$ where $E_1 \in E \cap (S/_{\mathcal{R}} \times S_1)$ and $F_1 \in F \cap (S/_{\mathcal{R}} \times S_1)$. We know that $Pbr(x_0', E_1, F_1) = 1$, i.e., $E_1 \rightarrow_\bullet F_1$ where $\rightarrow_\bullet$ is the transition relation on $S/_{\mathcal{R}} \times S_1$ iff $\forall s, s' \in Pred(E_1) \cap (S \times S_1)$, $Pbr(s, E_1, F_1) = Pbr(s', E_1, F_1) = 1$. Similarly $Pbr(x_0'', E_1, F_1) = 1$ iff $\forall s, s' \in Pred(E_1) \cap (S \times S_1)$, $Pbr(s, E_1, F_1) = Pbr(s', E_1, F_1) = 1$. Thus either $Pbr(x_0', E, F) = Pbr(x_0'', E, F) = 1$ or $Pbr(x_0', E, F) = Pbr(x_0'', E, F) = 0$.

c) The successors of x_0', x_0'' belong to $(S \times S_1)$ and $(S/_{\mathcal{R}} \times S_1)$ respectively. We already know that: $Pbr(x_0'', E_1, F_1) = 1$, i.e., $E_1 \rightarrow_\bullet F_1$ iff $\forall s, s' \in Pred(E_1) \cap (S \times S_1)$, $Pbr(s, E_1, F_1) = Pbr(s', E_1, F_1) = 1$. Taking $x_0' = s$, we get $Pbr(x_0'', E_1, F_1) = Pbr(x_0', E_1, F_1)$.

■

A.2 Proofs of Chapter 4

Proof of Theorem 4.10

Proof: Let $\mathcal{D} = (S, P, AP, L, s_0)$ be a DTMC and $\mathcal{D}/_{\mathcal{R}} = (S/_{\mathcal{R}}, P', AP, L', s_0')$ be its quotient under WPE. Since we have defined the WPE relation on a single state space, to prove this theorem we take the disjoint union $S \cup S/_{\mathcal{R}}$. Let us define a relation $\mathcal{R}' \subseteq (S \cup S/_{\mathcal{R}}) \times (S \cup S/_{\mathcal{R}})$ such that $\mathcal{R}' = \{(s, C) | s \in C, C \in S/_{\mathcal{R}}\}$. Let $\mathcal{R}^*$ be the reflexive, symmetric and transitive of closure of $\mathcal{R}'$.

Now we prove that $\mathcal{R}^*$ is a WPE relation. This is done by checking both conditions of Def. 4.4. Let $(s, C) \in \mathcal{R}^*$. The proofs for pairs (s, s'), (C, s), and (C, C) are similar and omitted.

1. $L'(C) = L(s)$ by definition of $\mathcal{D}/_{\mathcal{R}}$.

2. Next we prove that $\forall E, F \in (S \cup S/_{\mathcal{R}})/_{\mathcal{R}^*}$ and $\forall x_0', x_0'' \in Pred(E)$ it holds $wp(x_0', E, F) = wp(x_0'', E, F)$. Let $x_0', x_0'' \in Pred(E)$. Consider the following three cases based on the successors of x_0', x_0'' such that these successors are in E.

a) The successors of both x'_0, x''_0 belong to S. Since we know that $\mathcal{R}$ is a WPE, it follows $wp(x'_0, E, F) = wp(x''_0, E, F)$.

b) The successors of both x'_0, x''_0 belong to $S/_\mathcal{R}$. In this case, $wp(x'_0, E, F) = wp(x'_0, E_1, F_1)$ where $E_1 \in E \cap S/_\mathcal{R}$ and $F_1 \in F \cap S/_\mathcal{R}$, which equals

$$\sum_{x' \in E_1} \frac{P(x'_0, x')}{P(x'_0, E_1)} \cdot P'(x', F_1) = P'(E_1, F_1).$$

Similarily $wp(x''_0, E, F) = wp(x''_0, E_1, F_1) = P'(E_1, F_1)$.

c) The successors of x'_0, x''_0 belong to S and $S/_\mathcal{R}$ respectively. In this case we get, $wp(x''_0, E, F) = wp(x''_0, E_1, F_1) = P'(E_1, F_1)$. From Def. 4.6 we know that:

$$P'(E_1, F_1) = wp(x'_0, E_1, F_1) = wp(x'_0, E, F).$$

Since all the conditions of Def. 4.4 are satisfied by the relation $\mathcal{R}^*$, it is a WPE relation. ■

Proof of Lemma 4.13

Proof: Let $s_1 \sim s_2$. We prove that both conditions for WPE are satisfied.

- $L(s_1) = L(s_2)$, follows directly from $s_1 \sim s_2$.
- Let $C, D \in S/_\sim$ and $s'_0, s''_0 \in Pred(C)$. Since $P(s_1, D) = P(s_2, D)$ for all $s_1, s_2 \in C$, then for all $s^* \in C$:

$$\begin{aligned}
wp(s'_0, C, D) &= \sum_{s \in C} \frac{P(s'_0, s)}{P(s'_0, C)} \cdot P(s, D) \\
&= P(s^*, D) \cdot \sum_{s \in C} \frac{P(s'_0, s)}{P(s'_0, C)} \\
&= P(s^*, D) \\
&= P(s^*, D) \cdot \sum_{s \in C} \frac{P(s''_0, s)}{P(s''_0, C)} \\
&= \sum_{s \in C} \frac{P(s''_0, s)}{P(s''_0, C)} \cdot P(s, D) \\
&= wp(s''_0, C, D).
\end{aligned}$$

Thus s_1, s_2 are WPE related. Consider the equivalence class $C = \{s_4, s_5, s_6\}$ under WPE $\mathcal{R}$ in Fig. 4.2 (left). Here $s_4 \not\sim s_5$ since s_4 can reach a c-state while

s_5 cannot. Thus we can conclude that $\sim$ is strictly finer than WPE. ■

Proof of Theorem 4.25

Proof: In order to prove this theorem it is sufficient to show that for each accepting cylinder Cyl set in $\mathcal{D}/_R$, there is a corresponding *set* of cylinder sets in the DTMC $\mathcal{D}$ that are accepted by the DRA $\mathcal{A}$ and that jointly have the same probability as Cyl. Consider the set Π of cylinder sets in $\mathcal{D}$, and $\mathcal{D}/_R$ that are accepted by DRA $\mathcal{A}$, s.t. $\forall Cyl = (s_0, s_1, ..., s_n)$, and $Cyl' = (s'_0, s'_1, ..., s'_n)$ with $s_i \stackrel{\circ}{=} s'_i$, $\forall 0 \leq i \leq n$ implies $Cyl' \in \Pi$. Then we have to prove,

$$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_n) = \Pr_D(\Pi_n). \tag{A.1}$$

We will prove this equation by induction over the length of the cylinder set Cyl.

- **Base Case**: In this case, $n = 0$ and

  $$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_0) = 1 = \Pr_D(\Pi_0),$$

  if $s_0 \in D, \Pi_0$, and 0, otherwise.

- **Induction Hypothesis**: Assume that for cylinder sets of length $n \in \mathbb{N}$, it holds:

  $$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_n) = \Pr_D(\Pi_n).$$

- **Induction Step**: Consider the case $n + 1$:

  $$\begin{aligned}
  &\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_{n+1}) \\
  &= \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{s_2 \in S} P(s_1, s_2) \cdot \Pr_{s_2}(\Pi_n) \\
  &= \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{C \in S/_{\mathcal{R}}} \sum_{s_2 \in C} P(s_1, s_2) \cdot \Pr_{s_2}(\Pi_n) \\
  &= \sum_{C \in S/_{\mathcal{R}}} \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{s_2 \in C} P(s_1, s_2) \cdot \Pr_{s_2}(\Pi_n).
  \end{aligned}$$

  Multiplying the above expression by $\dfrac{P(s_1, C)}{P(s_1, C)}$ we get:

$$\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s'_0,s_1,D)\cdot\sum_{s_2\in D}\frac{P(s_1,C)}{P(s_1,C)}\cdot P(s_1,s_2)\cdot\Pr_{s_2}(\Pi_n)$$
$$=\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s'_0,s_1,D)\cdot P(s_1,C)\cdot\sum_{s_2\in C}\frac{P(s_1,s_2)}{P(s_1,C)}\cdot\Pr_{s_2}(\Pi_n)$$
$$=\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s'_0,s_1,D)\cdot P(s_1,C)\cdot\sum_{s_2\in C} P(s_1,s_2,C)\cdot\Pr_{s_2}(\Pi_n).$$

From the induction hypothesis we have:

$$\sum_{s_2\in C} P(s_1,s_2,C)\cdot\Pr_{s_2}(\Pi_n)=\Pr_{C}(\Pi_n).$$

Also from Def. 4.3 and Def. 4.6 we know that:

$$\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s'_0,s_1,D)\cdot P(s_1,C)=\sum_{C\in S/_{\mathcal{R}}} P'(D,C),$$

since $\sum_{s_1\in D} P(s'_0,s_1,D)\cdot P(s_1,C)=wp(s'_0,D,C)=P'(D,C)$. Therefore we get:

$$\sum_{C\in S/_{\mathcal{R}}} P'(D,C)\cdot\Pr_{C}(\Pi_n)=\Pr_{D}(\Pi_{n+1}).$$

■

Proof of Theorem 4.30

Proof: Let $\mathcal{D}=(S,P,AP,L,s_0)$ be a DTMC and $\mathcal{D}/_{\mathcal{R}}=(S/_{\mathcal{R}},P',AP,L',s'_0)$ be its quotient under WPE $\mathcal{R}$. Let $\mathcal{D}_1=(S_1,P_1,AP_1,L_1,s_p)$ be a DTMC composed in parallel to $\mathcal{D}$ and $\mathcal{D}/_{\mathcal{R}}$, i.e., $\mathcal{D}\otimes\mathcal{D}_1$, $\mathcal{D}/_{\mathcal{R}}\otimes\mathcal{D}_1$. Since we have defined WPE relation on a single state space, to prove this theorem we take the disjoint union $(S\times S_1)\cup(S/_{\mathcal{R}}\times S_1)$. Let us define a relation $\mathcal{R}'\subseteq((S\times S_1)\cup(S/_{\mathcal{R}}\times S_1))\times((S\times S_1)\cup(S/_{\mathcal{R}}\times S_1))$ such that $\mathcal{R}'=\{((s,t),(C,t))|s\in C,t\in S_1\}$, $C\in S/_{\mathcal{R}}$. Let $\mathcal{R}^*$ be the the reflexive, symmetric and transitive closure of $\mathcal{R}'$
Now we prove that $\mathcal{R}^*$ is a WPE relation. This is done by checking both conditions of Def. 4.4. Let $((s,t),(C,t))\in\mathcal{R}^*$.

1. From the definition of $\mathcal{D}/_{\mathcal{R}}$ we know that, $L'(C)=L(s)$. Nowe we have to show that: $L(s,t)=L(C,t)$. From Def. 4.29 we know that $L(s,t)=L(s)\cup L(t)=L(C)\cup L(t)=L(C,t)$.

2. Next we prove that $\forall E, F \in ((S \times S_1) \cup (S/_{\mathcal{R}} \times S_1))/_{\mathcal{R}^*}$ and $\forall x'_0, x''_0 \in Pred(E)$ it holds $wp(x'_0, E, F) = wp(x''_0, E, F)$. Let $x'_0, x''_0 \in Pred(E)$. Consider the following three cases based on the successors of x'_0, x''_0 such that these successors are in E.

   a) The successors of both x'_0, x''_0 belong to $S \times S_1$. Since $C \in S/_{\mathcal{R}}$ we know that $wp(s'_0, C, D) = wp(s''_0, C, D)$, where $s'_0, s''_0 \in Pred(C)$. Since parallel composition is the synchronous product, where two DTMCs move in a lock step fashion, i.e., $\forall s_1, s_2 \in C$ and $t \in S_1$, $\frac{s_1 \xrightarrow{p_1} s_3 \wedge s_2 \xrightarrow{p_2} s_4 \wedge t \xrightarrow{p_3} t'}{(s_1,t) \xrightarrow{p_1 \cdot p_3} (s_3,t') \wedge (s_2,t) \xrightarrow{p_2 \cdot p_3} (s_4,t')}$. That is all the transitions from an equivalence class, say from $C \in S/_{\mathcal{R}}$ to $D \in S/_{\mathcal{R}}$ get multiplied by the same factor in the product DTMC and therefore $\forall (s_1,t), (s_2,t) \in S \times S_1$ s.t. $s_1, s_2 \in C$ it holds: $wp(x'_0, E, F) = wp(x''_0, E, F)$.

   b) The successors of both x'_0, x''_0 belong to $(S/_{\mathcal{R}} \times S_1)$. We have $wp(x'_0, E, F) = wp(x'_0, E_1, F_1)$ where $E_1 \in E \cap (S/_{\mathcal{R}} \times S_1)$ and $F_1 \in F \cap (S/_{\mathcal{R}} \times S_1)$, which equals

$$\sum_{x' \in E_1} \frac{P'_s(x'_0, x')}{P'_s(x'_0, E_1)} \cdot P'_s(x', F_1) = P'_s(E_1, F_1),$$

   where P'_s is the probability matrix for $\mathcal{D}/_{\mathcal{R}} \otimes D_1$ defined on $(S/_R \times S_1)$. Similarily $wp(x''_0, E, F) = wp(x''_0, E_1, F_1) = P'_s(E_1, F_1)$.

   c) The successors of x'_0, x''_0 belong to $(S \times S_1)$ and $(S/_{\mathcal{R}} \times S_1)$ respectively. We already know that:

$$wp(x''_0, E, F) = wp(x''_0, E_1, F_1) = P'_s(E_1, F_1).$$

   From Def. 4.6 we know that $wp(s_0, C, D) = P(C, D)$, where $s_0 \in Pred(C)$, and $C, D \in S/_{\mathcal{R}}$, therefore while taking the parallel composition, corresponding transitions of both DTMCs $\mathcal{D}, \mathcal{D}/_{\mathcal{R}}$ get multiplied by the same factor, i.e., $\forall s \in C$ and $t \in S_1$, $\frac{t \xrightarrow{p_3} t' \wedge s \xrightarrow{p_1} s' \wedge C \xrightarrow{p_2} D}{(s,t) \xrightarrow{p_1 \cdot p_3} (s',t') \wedge (C,t) \xrightarrow{p_2 \cdot p_3} (D,t')}$, so as to obtain $(\mathcal{D} \otimes \mathcal{D}_1), (\mathcal{D}/_{\mathcal{R}} \otimes \mathcal{D}_1)$. Thus $P'_s(E_1, F_1) = wp(x'_0, E_1, F_1) = wp(x'_0, E, F)$.

■

Proof of Theorem. 4.40

Proof: Since the class of languages accepted by DRAs agrees with the class of ω-regular languages, therefore any property that can be expressed using LTL

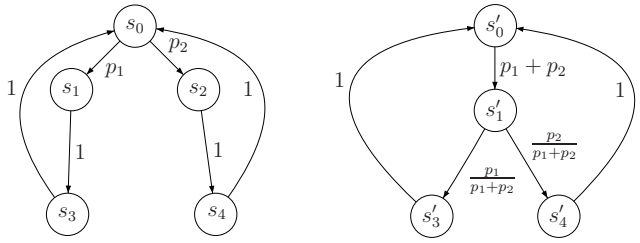

Figure A.1: Stationary state probabilities in DTMC

can also be expressed using DRA. From the proof of Thm. 4.25 we know that for any cylinder set Cyl in $\mathcal{D}/_R$ which satisfies the LTL formula φ, there is a corresponding set of cylinder sets in $\mathcal{D}$ that satisfy φ and that jointly have the same probability as Cyl. It is also easy to check that the cumulative reward earned by each of these cylinder sets is same (since in the reward setting, WPE related states need to have the same reward). Hence we can conclude:

$$\mathcal{E}(\mathcal{MD} \models \varphi) = \mathcal{E}(\mathcal{MD}/_R \models \varphi).$$

■

Proof of Theorem 4.42

Proof: From Corollary 4.28 we know that WPE preserves transient-state probabilities, i.e., $\sum_{s \in C} \mathcal{T}(s_0, s, n) = \mathcal{T}(s'_0, C, n)$. We also know that $rew(s) = rew(C)$ where $s \in C$. Now the proof is straightforward. ■

Proof of Theorem 4.44

Proof: In order to prove this theorem, it is sufficient to show that WPE preserves stationary state probabilities. We do this by showing that the system of linear equations for the original DTMC on the left (Fig. A.1) can be transformed into a system of linear equations having the same number of variables, equations and coefficient matrix as the linear system for the quotient DTMC on the right (Fig. A.1). Let $\mathcal{S}_l(s_i)$, and $\mathcal{S}_r(s'_i)$ be the stationary probabilities of being in state s_i, s'_i in the left and right DTMCs respectively. The system of linear equations for the original DTMC in the Fig. A.1 given above is given as:

$$\mathcal{S}_l(s_0) \cdot p_1 = \mathcal{S}_l(s_1) \cdot 1 \tag{A.2}$$

$$\mathcal{S}_l(s_0) \cdot p_2 = \mathcal{S}_l(s_2) \cdot 1 \tag{A.3}$$

$$\mathcal{S}_l(s_1) \cdot 1 = \mathcal{S}_l(s_3) \cdot 1 \tag{A.4}$$

$$\mathcal{S}_l(s_2) \cdot 1 = \mathcal{S}_l(s_4) \cdot 1 \tag{A.5}$$

$$\mathcal{S}_l(s_3) \cdot 1 + \mathcal{S}_l(s_4) \cdot 1 = \mathcal{S}_l(s_0) \cdot (p_1 + p_2) \tag{A.6}$$

For the quotient DTMC the linear system is gives as:

$$\mathcal{S}_r(s'_0) \cdot (p_1 + p_2) = \mathcal{S}_r(s'_1) \cdot (\frac{p_1}{p_1 + p_2} + \frac{p_2}{p_1 + p_2}) \tag{A.7}$$

$$\mathcal{S}_r(s'_1) \cdot \frac{p_1}{p_1 + p_2} = \mathcal{S}_r(s'_3) \cdot 1 \tag{A.8}$$

$$\mathcal{S}_l(s'_1) \cdot \frac{p_2}{p_1 + p_2} = \mathcal{S}_r(s'_4) \cdot 1 \tag{A.9}$$

$$\mathcal{S}_r(s'_3) \cdot 1 + \mathcal{S}_r(s'_4) \cdot 1 = \mathcal{S}_r(s'_0) \cdot (p_1 + p_2) \tag{A.10}$$

By taking the sum of Eqn. A.2 and A.3 we get:

$$\mathcal{S}_l(s_0) \cdot (p_1 + p_2) = \mathcal{S}_l(s_1) \cdot 1 + \mathcal{S}_l(s_2) \cdot 1$$

Let $z = \mathcal{S}_l(s_1) + \mathcal{S}_l(s_2)$, then the above equation can be rewritten as:

$$\mathcal{S}_l(s_0) \cdot (p_1 + p_2) = z \cdot 1$$

From this we can see that:

$$\mathcal{S}_l(s_0) = \frac{z \cdot 1}{(p_1 + p_2)} \tag{A.11}$$

From this we also get Eqn. A.12, A.13:

$$\mathcal{S}_l(s_1) = \frac{z \cdot 1}{(p_1 + p_2)} \cdot p_1 \tag{A.12}$$

$$\mathcal{S}_l(s_2) = \frac{z \cdot 1}{(p_1 + p_2)} \cdot p_2 \tag{A.13}$$

Eqn. A.4 and A.5 can now be rewritten as:

$$\mathcal{S}_l(s_3) \cdot 1 = \frac{z \cdot 1}{(p_1 + p_2)} \cdot p_1 \tag{A.14}$$

$$\mathcal{S}_l(s_4) \cdot 1 = \frac{z \cdot 1}{(p_1 + p_2)} \cdot p_2 \tag{A.15}$$

Now these system of equations have the same form as that of quotient DTMC, where we have:

$$\mathcal{S}_l(s_0) = \mathcal{S}_r(s'_0) \tag{A.16}$$

$$z = \mathcal{S}_r(s'_1) \tag{A.17}$$

$$\mathcal{S}_l(s_3) = \mathcal{S}_r(s'_3) \tag{A.18}$$

$$\mathcal{S}_l(s_4) = \mathcal{S}_r(s'_4) \tag{A.19}$$

Since $z = \mathcal{S}_l(s_1) + \mathcal{S}_l(s_2) = \mathcal{S}_r(s'_1)$ we can conclude that stationary state probabilities are preserved. ■

This proof can be easily extended to any arbitrary DTMC $\mathcal{D}$ with finite number of states, where the general system of linear equations for left hand and right hand side is given as:

$$\sum_{s\in B, s\neq s'} \mathcal{S}_l(s) \cdot P(s,s') = \mathcal{S}_l(s') \cdot \sum_{s\in B, s\neq s'} P(s',s)$$

where $\sum_{s\in B} \mathcal{S}_l(s) = 1$.

$$\sum_{C\in B', C\neq D} \mathcal{S}_r(C) \cdot P'(C,D) = \mathcal{S}_r(D) \cdot \sum_{C\in B', C\neq D} P'(D,C)$$

where $\sum_{C\in B'} \mathcal{S}_r(C) = 1$, and $s \in C$, $C, D \in S/_{\mathcal{R}}$.

A.3 Proofs of Chapter 5

Proof of Theorem 5.9

Proof: Let $\mathcal{C} = (S, R, AP, L, s_0)$ be a CTMC and $\mathcal{C}/_{\mathcal{R}} = (S/_{\mathcal{R}}, R', AP, L', s'_0)$ be its quotient under WL. Since we have defined the WL relation on a single state space, to prove this theorem we take the disjoint union $S \cup S/_{\mathcal{R}}$. Let us define an equivalence relation $\mathcal{R}^* \subseteq (S \cup S/_{\mathcal{R}}) \times (S \cup S/_{\mathcal{R}})$ with $\{(s, C) | s \in C\} \subseteq \mathcal{R}^*$. The exit rate $E'(C)$ for $C \in S/_{\mathcal{R}}$ is defined by $\sum_{x\in(S\cup S/_{\mathcal{R}})} R'(C,x)$.

Now we prove that $\mathcal{R}^*$ is a WL relation. This is done by checking both conditions of Def. 5.4. Let $(s, C) \in \mathcal{R}^*$. The proofs for pairs (s, s'), (C, s), and (C, C) are similar and omitted.

1. $L'(C) = L(s)$ by definition of $\mathcal{C}/_{\mathcal{R}}$. We prove that $E'(C) = E(s)$ as follows:

$$\begin{aligned}
E'(C) &= \sum_{x \in (S \cup S/_{\mathcal{R}})} R'(C,x) = \sum_{D \in S/_{\mathcal{R}}} R'(C,D) \\
&= \sum_{D \in S/_{\mathcal{R}}} wr(s'_0, C, D) \text{ for some } s'_0 \in Pred(C) \\
&= \sum_{D \in S/_{\mathcal{R}}} \sum_{s \in C} P(s'_0, s, C) \cdot R(s, D) \\
&= \sum_{s \in C} \left(P(s'_0, s, C) \cdot \sum_{D \in S/_{\mathcal{R}}} R(s, D) \right) \\
&= \sum_{s \in C} \left(P(s'_0, s, C) \cdot \sum_{D \in S/_{\mathcal{R}}} \sum_{s' \in D} R(s, s') \right) \\
&= \sum_{s \in C} \left(P(s'_0, s, C) \cdot \sum_{s' \in S} R(s, s') \right) \\
&= \sum_{s \in C} (P(s'_0, s, C) \cdot E(s)) \\
&= \left(\sum_{s \in C} P(s'_0, s, C) \right) \cdot E(s), \text{ since for all } s' \in C, E(s') = E(s) \\
&= E(s).
\end{aligned}$$

2. Finally we prove that $\forall E, F \in (S \cup S/_{\mathcal{R}})/_{\mathcal{R}^*}$ and $\forall x'_0, x''_0 \in Pred(E)$ it holds $wr(x'_0, E, F) = wr(x''_0, E, F)$. Let $x'_0, x''_0 \in Pred(E)$. Consider the following three cases based on the successors of x'_0, x''_0 such that these successors are in E.

   a) The successors of both x'_0, x''_0 belong to S. Since we know that $\mathcal{R}$ is a WL, it follows $wr(x'_0, E, F) = wr(x''_0, E, F)$.

   b) The successors of both x'_0, x''_0 belong to $S/_{\mathcal{R}}$. In this case, $wr(x'_0, E, F) = wr(x'_0, \{E_1\}, F)$ where $E_1 \in E \cap S/_{\mathcal{R}}$, which equals

$$\sum_{x' \in \{E_1\}} \frac{P(x'_0, x')}{P(x'_0, E_1)} \cdot R'(x', F) = R'(E_1, F).$$

   Similarily $wr(x''_0, E, F) = wr(x''_0, \{E_1\}, F) = R'(E_1, F)$.

c) The successors of x'_0, x''_0 belong to S and $S/_{\mathcal{R}}$ respectively. In this case we get, $wr(x''_0, E, F) = wr(x''_0, \{E_1\}, F) = R'(E_1, F)$.

We know that the successors of $E_1 \in S/_{\mathcal{R}}$, hence using Def. 5.6 we conclude:

$$R'(E_1, F) = wr(x'_0, E_1, F) = wr(x'_0, E, F).$$

Since all the conditions of Def. 5.4 are satisfied by the relation $\mathcal{R}^*$, it is a WL relation. ■

Proof of Lemma 5.12

Proof: Let $s_1 \sim s_2$. We prove that both conditions for WL are satisfied.

- $L(s_1) = L(s_2)$, follows directly from $s_1 \sim s_2$.
- $E(s_1) = E(s_2)$, since we know that

$$E(s_1) = \sum_{s \in S} R(s_1, s) = \sum_{C \in S/_\sim} \sum_{s \in C} R(s_1, s) = \sum_{C \in S/_\sim} R(s_1, C).$$

If $s_1 \sim s_2$, then $R(s_1, C) = R(s_2, C)$. Therefore:

$$E(s_1) = \sum_{C \in S/_\sim} R(s_1, C) = \sum_{C \in S/_\sim} R(s_2, C) = E(s_2).$$

- Let $C, D \in S/_\sim$ and $s'_0, s''_0 \in Pred(C)$. Since $R(s_1, D) = R(s_2, D)$ for all $s_1, s_2 \in C$, then for all $s^* \in C$:

$$\begin{aligned}
wr(s'_0, C, D) &= \sum_{s \in C} \frac{P(s'_0, s)}{P(s'_0, C)} \cdot R(s, D) \\
&= R(s^*, D) \cdot \sum_{s \in C} \frac{P(s'_0, s)}{P(s'_0, C)} \\
&= R(s^*, D) \\
&= R(s^*, D) \cdot \sum_{s \in C} \frac{P(s''_0, s)}{P(s''_0, C)} \\
&= \sum_{s \in C} \frac{P(s''_0, s)}{P(s''_0, C)} \cdot R(s, D) \\
&= wr(s''_0, C, D).
\end{aligned}$$

Thus s_1, s_2 are WL related. ■

Proof of Lemma 5.22

Proof: We will prove this lemma by induction over the length of the cylinder set $Cyl \in \Pi$. That is, we will prove for any $n \in \mathbb{N}$:

$$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_n) = \Pr_{D}(\Pi_n).$$

- **Base Case**: In this case, $n = 0$ and

  $$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_0) = 1 = \Pr_{D}(\Pi_0),$$

  if $s_0 \in D, \Pi_0$, and 0, otherwise.

- **Induction Hypothesis**: Assume that for cylinder sets of length $n \in \mathbb{N}$, it holds:

  $$\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_n) = \Pr_{D}(\Pi_n).$$

- **Induction Step**: Consider the case $n + 1$:

  $$\begin{aligned}
  &\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \Pr_{s_1}(\Pi_{n+1}) \\
  &= \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{s_2 \in S} P(s_1, s_2) \cdot (e^{-E(s_1) \cdot \inf I_0} - e^{-E(s_1) \cdot \sup I_0}) \cdot \Pr_{s_2}(\Pi_n)
  \end{aligned}$$

  Let $(e^{-E(s_1) \cdot \inf I_0} - e^{-E(s_1) \cdot \sup I_0}) = \delta(s_1, I_0)$, then the above expression is equal to:

  $$\begin{aligned}
  &\sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{s_2 \in S} P(s_1, s_2) \cdot \delta(s_1, I_0) \cdot \Pr_{s_2}(\Pi_n) \\
  &= \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{C \in S/\mathcal{R}} \sum_{s_2 \in C} P(s_1, s_2) \cdot \delta(s_1, I_0) \cdot \Pr_{s_2}(\Pi_n) \\
  &= \sum_{C \in S/\mathcal{R}} \sum_{s_1 \in D} P(s'_0, s_1, D) \cdot \sum_{s_2 \in C} P(s_1, s_2) \cdot \delta(s_1, I_0) \cdot \Pr_{s_2}(\Pi_n).
  \end{aligned}$$

  Multiplying the above expression by $\dfrac{R(s_1, C)}{R(s_1, C)}$ and using $P(s_1, s_2) =$

$\frac{R(s_1,s_2)}{E(s_1)}$ yields:

$$\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s_0',s_1,D)\cdot\sum_{s_2\in D}\frac{R(s_1,C)}{R(s_1,C)}\cdot\frac{R(s_1,s_2)}{E(s_1)}\cdot\delta(s_1,I_0)\cdot\Pr_{s_2}(\Pi_n).$$

Since $\forall s_1,s_1'\in D,\ E(s_1)=E(s_1')$, we have $\delta(s_1,I_0)=\delta(s_1',I_0)$. We get:

$$\frac{\delta(s_1,I_0)}{E(s_1)}\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s_0',s_1,D)\cdot R(s_1,C)\cdot\sum_{s_2\in C}\frac{R(s_1,s_2)}{R(s_1,C)}\cdot\Pr_{s_2}(\Pi_n)$$

$$=\frac{\delta(s_1,I_0)}{E(s_1)}\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s_0',s_1,D)\cdot R(s_1,C)\cdot\sum_{s_2\in C}P(s_1,s_2,C)\cdot\Pr_{s_2}(\Pi_n).$$

We have already proved that $\forall s\in D,\ E(s)=E(D)$, cf. Thm. 5.9. From the induction hypothesis we have:

$$\sum_{s_2\in C}P(s_1,s_2,C)\cdot\Pr_{s_2}(\Pi_n)=\Pr_{C}(\Pi_n).$$

Also from Def. 5.1 and Def. 5.6 we know that:

$$\sum_{C\in S/_{\mathcal{R}}}\sum_{s_1\in D} P(s_0',s_1,D)\cdot R(s_1,C)=\sum_{C\in S/_{\mathcal{R}}}R'(D,C),$$

since $\sum_{s_1\in D}P(s_0',s_1,D)\cdot R(s_1,C)=wr(s_0',D,C)=R'(D,C)$. Therefore we get:

$$\frac{\delta(D,I_0)}{E(D)}\sum_{C\in S/_{\mathcal{R}}}R'(D,C)\cdot\Pr_{C}(\Pi_n)=\Pr_{D}(\Pi_{n+1}).$$

■

Proof of Theorem 5.21

Proof: Let C_n be the set of all the cylinder sets in $\mathcal{C}$, and $\mathcal{C}/_{\mathcal{R}}$ of length n that are accepted by DTA $\mathcal{A}$ and $C_{n/\Pi}$ be the set of subsets of C_n grouped according

to WL-closed set of cylinder sets. Let Cyl_π be the cylinder set that contains π. Since the cylinder sets in Eq. 5.1 are disjoint, we have:

$$\begin{aligned}\Pr(\mathcal{C} \models \mathcal{A}) &= \Pr\left(\bigcup_{n\in\mathbb{N}} \bigcup_{\pi\in Paths_n^{\mathcal{C}}(\mathcal{A})} Cyl_\pi\right) \\ &= \sum_{n\in\mathbb{N}} \sum_{Cyl\in C_n} \Pr(Cyl) \\ &= \sum_{n\in\mathbb{N}} \sum_{\Pi\in C_{n/\Pi}} \sum_{D\in S/_R} \sum_{s_1\in D} P(s_0', s_1, D) \cdot \Pr_{s_1}(\Pi).\end{aligned}$$

Then we get using Eq. 5.2:

$$\begin{aligned}\Pr(\mathcal{C} \models \mathcal{A}) &= \sum_{n\in\mathbb{N}} \sum_{\Pi\in C_{n/\Pi}} \sum_{D\in S/_{\mathcal{R}}} \sum_{s_1\in D} P(s_0', s_1, D) \cdot \Pr_{s_1}(\Pi) \\ &= \sum_{n\in\mathbb{N}} \sum_{\Pi\in C_{n/\Pi}} \sum_{D\in S/_{\mathcal{R}}} \Pr_{D}(\Pi) \\ &= \Pr(\mathcal{C}/_{\mathcal{R}} \models \mathcal{A}).\end{aligned}$$

■

Proof of Theorem. 5.28

Proof: We prove the measurability by showing that for any path $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \in Paths_n^{\mathcal{C}}(s_0 \models \varphi)$ where $Paths_n^{\mathcal{C}}(s_0 \models \varphi)$ is the set of paths of length n starting from s_0 that satisfy φ, there exists a cylinder set $Cyl(s_0, I_0, ..., I_{n-1}, s_n$ (Cyl for short) s.t. $\pi \in Cyl$ and $Cyl \subseteq Paths_n^{\mathcal{C}}(s_0 \models \varphi)$. Since the only interesting case is time-bounded "until", we consider $\varphi = \varphi_1 \mathrm{U}^{[a,b]} \varphi_2$, where $a, b \in \mathbb{Q}$. Let $\sum_{i=0}^{n-1} t_i - \Delta > a$ and $\sum_{i=0}^{n-2} t_i + \Delta < b$, where $\Delta = \dfrac{2n}{10^k}$, and k is large enough. We construct Cyl by considering intervals I_i with rational bounds that are based on t_i. Let $I_i = [t_i^-, t_i^+]$ s.t. $t_i^- = t_i = t_i^+$ if $t_i \in \mathbb{Q}$, and otherwise:

$$t_i^- < t_i < t_i^+,\ t_i^- > t_i - \frac{\Delta}{2n} \text{ and } t_i^+ < t_i + \frac{\Delta}{2n}.$$

We have to show for $t_i \notin \mathbb{Q}$, Eq. A.20 and Eq. A.21 hold:

$$\sum_{i=0}^{n-1} t_i^- > a. \tag{A.20}$$

Proof: We know that $\sum_{i=0}^{n-1} t_i - \Delta > a \implies \sum_{i=0}^{n-1} t_i^- + n \cdot \frac{\Delta}{2n} - \Delta > a$

$\implies \sum_{i=0}^{n-1} t_i^- + \frac{\Delta}{2} - \Delta > a \implies \sum_{i=0}^{n-1} t_i^- - \frac{\Delta}{2} > a \implies \sum_{i=0}^{n-1} t_i^- > a.$ ■

$$\sum_{i=0}^{n-2} t_i^+ < b. \tag{A.21}$$

Proof: We know that $\sum_{i=0}^{n-2} t_i + \Delta < b \implies \sum_{i=0}^{n-2} t_i^+ - (n-1) \cdot \frac{\Delta}{2n} + \Delta < b$

$\implies \sum_{i=0}^{n-2} t_i^+ + \frac{(n+1) \cdot \Delta}{2n} < b \implies \sum_{i=0}^{n-2} t_i^+ < b.$ ■

One way is to pick t_i^-, t_i^+ as follows:

$$t_i^- = \lfloor t_i \rfloor + \frac{\lfloor \{t_i\} \cdot 10^k \rfloor}{10^k},$$
$$t_i^+ = \lfloor t_i \rfloor + \frac{\lfloor \{t_i\} \cdot 10^k \rfloor + 1}{10^k},$$

where $\{t_i\}$ represents the fractional part of the irrational number t_i. It can be checked that picking t_i^-, t_i^+ this way satisfies the above mentioned constraints.

From this derivation we conclude that $\{\pi \in Paths(s_0) | \pi \models \varphi\}$ can be rewritten as the combination of cylinder sets of the form $Cyl = (s_0, I_0, ..., I_{n-1}, s_n)$. That is,

$$\{\pi \in Paths(s_0) | \pi \models \varphi\} = \bigcup_{n \in \mathbb{N}} \bigcup_{\pi \in Paths_n^{\mathcal{C}}(s_0 \models \varphi)} Cyl_\pi, \tag{A.22}$$

where $Paths_n^{\mathcal{C}}(s_0 \models \varphi)$ is the set of paths of length n starting from s_0 which satisfy φ. ■

Proof of Theorem. 5.30

Proof: The proof is similar to that of Thm. 5.21. We consider the WL-closed set of cylinder sets of length n in $\mathcal{C}$, $\mathcal{C}/_{\mathcal{R}}$ such that this set satisfies φ. The rest of the proof remains the same. ■

A.4 Proofs of Chapter 6

Proof of Theorem 6.20

Proof: We provide the proof of the law CCL-L. The proofs of the other CCL laws are similar. Let $\mathcal{T} = (\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_2$ and $\mathcal{U} = \mathcal{N}_1 \bullet (\mathcal{N}_2||\mathcal{M}_2)$. In order to prove that $\mathcal{T} \equiv \mathcal{U}$, we have to show $\mathcal{T} \preceq_S \mathcal{U}$ and $\mathcal{U} \preceq_S \mathcal{T}$. Let $((x,y),z)$ and $(x,(y,z))$ denote states of MTSs $\mathcal{T}$ and $\mathcal{U}$, respectively. Here x, y, z represents the state components of $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$, respectively. We only prove $\mathcal{T} \preceq_S \mathcal{U}$, the proof of $\mathcal{U} \preceq_S \mathcal{T}$ is very similar. To show $\mathcal{T} \preceq_S \mathcal{U}$, we have to prove that $\mathcal{T}$ strongly refines $\mathcal{U}$ according to Def. 6.5. Let S^* and S be the state space of $\mathcal{T}$ and $\mathcal{U}$. Let $\mathcal{R} \subseteq S^* \times S$ be a binary relation such that $\mathcal{R} = \{(((x,y),z),(x,(y,z))),(((x',y),z),(x',(y,z))),(((x,y'),z'),(x,(y',z'))) \ldots\}$. Intuitively $\mathcal{R}$ relates states of S^* to those states in S that have the same individual state components from $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$. Let us consider a state $s = (x,(y,z))$ from $\mathcal{U}$ and $s^* = ((x,y),z)$ from $\mathcal{T}$. From the definition of $\mathcal{R}$, we know that $(s^*, s) \in \mathcal{R}$. A transition from s in $\mathcal{U}$ or s^* in $\mathcal{T}$ can be performed by 1) $\mathcal{N}_1$ individually, or 2) $\mathcal{N}_2$ individually, or 3) $\mathcal{M}_2$ individually, or 4) $\mathcal{N}_1$ and $\mathcal{N}_2$ simultaneously, or 5) $\mathcal{N}_2$ and $\mathcal{M}_2$ simultaneously. We show that for every case[1] the conditions of Def. 6.5 are satisfied, and thus $\mathcal{R}$ is a strong refinement relation between $\mathcal{T}$ and $\mathcal{U}$. It is easy to check that condition 3 which requires final states to be related is trivially satisfied. This is because $\mathcal{R}$ relates states that have the same individual state components from $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$.

1) $\mathcal{N}_1$ individually: Let $s \xrightarrow{a}_\top s'$ (resp. $s \xrightarrow{a}_? s'$) be a transition of $\mathcal{N}_1$ taken in $\mathcal{U}$, where $s' = (x',(y,z))$. Since $\mathcal{N}_1$ is the left operand of layering in $\mathcal{U}$ such a transition is also possible from the related state $s^* = ((x,y),z)$, i.e., $s^* \xrightarrow{a}_\top s^{**}$ (resp. $s^* \xrightarrow{a}_? s^{**}$), where $s^{**} = ((x',y),z)$. From the definition of $\mathcal{R}$ we know that $(s^{**}, s') \in \mathcal{R}$. Similarly, it can be shown that for every transition $s^* \xrightarrow{a}_\top s^{***}$ (resp. $s^* \xrightarrow{a}_? s^{***}$), there exists a corresponding transition, $s \xrightarrow{a}_\top s''$ (resp. $s \xrightarrow{a}_? s''$) s.t. $(s^{***}, s'') \in \mathcal{R}$.

2) $\mathcal{N}_2$ individually: Similar to case 1 above.

3) $\mathcal{M}_2$ individually: Let $s \xrightarrow{a}_\top s'$ (resp. $s \xrightarrow{a}_? s'$) be a transition of $\mathcal{M}_2$ taken in $\mathcal{U}$, where $s' = (x,(y,z'))$. Since this transition is possible after layering operator in $\mathcal{U}$, it is also possible in $\mathcal{T}$ from the state related to s, i.e., s^* as this action is not waiting for some action of $\mathcal{N}_1$ to be executed, and $\mathcal{U}$ induces fewer interleavings

[1] Note that we do not consider the case where $\mathcal{N}_1$ and $\mathcal{M}_2$ move simultaneously. This is due to the fact that $\mathcal{N}_1 \ddagger \mathcal{M}_2$, and therefore they cannot have common actions (since the dependency relation is reflexive).

due to dominance of layered composition operator. Next, we consider a transition of $\mathcal{M}_2$ taken in $\mathcal{T}$ from state s^*. Since $\mathcal{N}_1 \ddagger \mathcal{M}_2$, a similar transition exists in $\mathcal{U}$ from the state related to s^*, i.e., s as putting $\mathcal{M}_2$ after the layering operator is not a problem as it is independent from $\mathcal{N}_1$.
4) $\mathcal{N}_1$ and $\mathcal{N}_2$ simultaneously: Let $s \xrightarrow{a}_{\top} s'$ (resp. $s \xrightarrow{a}_{?} s'$) be a transition in $\mathcal{U}$ as a result of a synchronization of $\mathcal{N}_1$ and $\mathcal{N}_2$ on action a, where $s' = (x', (y', z))$. Again as $\mathcal{U}$ induces fewer interleavings due to dominance of layered operator, a similar transition is possible in $\mathcal{T}$ from state related to s, i.e., s^*. Similarly, for any transition in $\mathcal{T}$ from s^*, a corresponding transition is enabled in $\mathcal{U}$ as $(\mathcal{N}_2||\mathcal{M}_2)$ will not block it (due to the fact that the synchronizing action is not waiting for an action from $\mathcal{M}_2$ to be executed since parallel composition does not respect dependencies).
5) $\mathcal{N}_2$ and $\mathcal{M}_2$ simultaneously: Let $s \xrightarrow{a}_{\top} \varphi$ (resp. $s \xrightarrow{a}_{?} s'$) be a transition in $\mathcal{U}$ as a result of a synchronization of $\mathcal{N}_2$ and $\mathcal{M}_2$ on action a, where $s' = (x, (y', z'))$. As this action is possible in $(\mathcal{N}_2||\mathcal{M}_2)$, it means that this action is not waiting for the execution of some actions from $\mathcal{N}_1$. It is therefore possible to take the same transition in $\mathcal{T}$ from s^* as $(\mathcal{N}_1 \bullet \mathcal{N}_2)$ will not block it. Similarly, for any transition in $\mathcal{T}$ from s^*, there will be a corresponding transition in $\mathcal{U}$. This is due to the fact this action is not waiting for actions of $\mathcal{N}_1$ to be executed as $\mathcal{N}_1 \ddagger \mathcal{M}_2$. ■

Proof of Theorem 6.22

Proof: ($\exists$ reachability): We provide the proof of $\mathcal{M} \models^{\exists} \Diamond S_f \Rightarrow \mathcal{M}^{\setminus sync} \models^{\exists} \Diamond S_f$. The proof of $\mathcal{M}^{\setminus sync} \models^{\exists} \Diamond S_f \Rightarrow \mathcal{M} \models^{\exists} \Diamond S_f$ is similar. Let $\mathcal{T} \in [\![\mathcal{M}]\!]$ be an LTS s.t. $\mathcal{T} \models \Diamond S_f$. We know that every finite path $\pi \in Paths^{S_f}_{fin}(\mathcal{T})$ is a realisation of some finite execution $\rho \in Exec^{S_f}_{fin}(\mathcal{M})$, and no finite path can be a realisation of more than one finite execution in $\mathcal{M}$ (since $\mathcal{M}$ is deterministic). Let η be the set of all such finite executions where $\eta \subseteq Exec^{S_f}_{fin}(\mathcal{M})$. From the definition of $\mathcal{M}^{\setminus sync}$ we know that for each finite execution $\rho \in \eta$ there exists a corresponding finite execution in $Exec^{S_f}_{fin}(\mathcal{M}^{\setminus sync})$ obtained by allowing interleaving on common actions s.t. common action of $\mathcal{M}_1$ is executed first followed by execution of common action of $\mathcal{M}_2$. Let $\eta' \subseteq Exec^{S_f}_{fin}(\mathcal{M}^{\setminus sync})$ be the set of all such finite executions. Let $\mathcal{T}' \in [\![\mathcal{M}^{\setminus sync}]\!]$ be an LTS s.t. for every finite execution $\rho' \in \eta'$ there exists a finite path π' in $\mathcal{T}'$ that is a realisation of ρ' and $\mathcal{T}'$ does not contain any path $\pi' \in Paths^{S_f}_{fin}(\mathcal{T}')$ which is not a realisation of some finite execution $\rho' \in \eta'$. In other words we have constructed an implementation $\mathcal{T}'$ corresponding to $\mathcal{T}$ s.t. $\mathcal{T}' \models \Diamond S_f$.
($\forall$ reachability): We provide the proof of $\mathcal{M} \models^{\forall} \Diamond S_f \Rightarrow \mathcal{M}^{\setminus sync} \models^{\forall} \Diamond S_f$. The proof of $\mathcal{M}^{\setminus sync} \models^{\forall} \Diamond S_f \Rightarrow \mathcal{M} \models^{\forall} \Diamond S_f$ is similar. From the definition of $\forall$

reachability (Def. 6.11) we know that $\mathcal{M}$ reaches S_f if and only if all the implementations of $\mathcal{M}$ are able to reach S_f. This intuitively means that $\mathcal{M}$ does not have those *may* transitions that block any of its implementations from reaching the final state. Since $\mathcal{M}^{\setminus sync}$ is obtained from $\mathcal{M}$ by allowing interleaving on common actions, such *may* transitions (or equivalent transition sequences) are also absent in $\mathcal{M}^{\setminus sync}$. In other words, every implementation $\mathcal{T}' \in [\![\mathcal{M}^{\setminus sync}]\!]$ reaches S_f, i.e., $\mathcal{T}' \models \Diamond S_f$. ■

Proof of Lemma 6.25

Proof: From the definition of layered normal form (Def. 6.24) we know that all finite executions that are in LNF consists of the consecutive execution of actions of $\mathcal{M}_1$, followed by the consecutive execution of actions of $\mathcal{M}_2$. We know that in $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$, an action of $\mathcal{M}_2$ occurs only when all the actions in $\mathcal{M}_1$ on which it is dependent have been executed. This intuitively means that all the actions of $\mathcal{M}_2$ that occur in a finite execution before any action of $\mathcal{M}_1$ are independent of this action and thus by repeated permutation of these actions any finite execution $\rho \in Exec_{fin}^{S_f}((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ can be converted to a finite execution that is in LNF. ■

Proof of Theorem 6.27

Proof: Let $\mathcal{M}_1, \mathcal{M}_2$ be MTSs. From the definition of LNF (Def. 6.24), we know that a finite execution in LNF involves the consecutive execution of actions of $\mathcal{M}_1$, followed by the consecutive execution of actions of $\mathcal{M}_2$. This means that for every finite execution $\rho \in Exec_{fin}^{LNF}((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ there exists a finite execution ρ' in $((\mathcal{M}_1; \mathcal{M}_2)^{\setminus sync})$ that ends in the final state and where: $\forall n \geq 0 : \rho[n] \approx \rho'[n] \wedge \rho[n] \xrightarrow{a_{n+1}}_{\top} \rho[n+1] \Rightarrow \rho'[n] \xrightarrow{a_{n+1}}_{\top} \rho'[n+1]$(resp. $\rho[n] \xrightarrow{a_{n+1}}_{?} \rho[n+1] \Rightarrow \rho'[n] \xrightarrow{a_{n+1}}_{?} \rho'[n+1]$). The relation $\approx$ between states of $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ and $((\mathcal{M}_1; \mathcal{M}_2)^{\setminus sync})$ is defined as follows: $S_1 \times S_2$ is the state space of $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ and $(S_1 \setminus \{s_{f1}\} \cup S_2)$ is the state space of $((\mathcal{M}_1; \mathcal{M}_2)^{\setminus sync})$ then $\forall s_1 \in S_1$ where $s_1 \neq s_{f1} : (s_1, s_{02}) \approx s_1$ and $\forall s_2 \in S_2 : (s_{f1}, s_2) \approx s_2$.

In other words we have related every finite execution of $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ that is in LNF to some finite execution in $((\mathcal{M}_1; \mathcal{M}_2)^{\setminus sync})$ that ends in the final state and vice-versa. It is also clear from Lemma 6.25 that for every finite execution ρ in $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ that ends in the final state, there exists a finite execution ρ' in $((\mathcal{M}_1 \bullet \mathcal{M}_2)^{\setminus sync})$ that is LNF s.t. $\rho \equiv_{po}^{*} \rho'$. ■

Proof of Theorem 6.29

Proof: ($\exists$ reachability): We provide the proof of $\mathcal{M} \models^{\exists} \Diamond S_f \Rightarrow \mathcal{M}' \models^{\exists} \Diamond S'_f$. The proof of $\mathcal{M}' \models^{\exists} \Diamond S'_f \Rightarrow \mathcal{M} \models^{\exists} \Diamond S_f$ is similar. Let $\mathcal{T} \in [\![\mathcal{M}^{\setminus sync}]\!]$ be an LTS s.t. $\mathcal{T} \models \Diamond S_f$. We know that every finite path $\pi \in Paths_{fin}^{S_f}(\mathcal{T})$ is a realisation of

some finite execution $\rho \in Exec^{S_f}_{fin}(\mathcal{M}^{\setminus sync})$, and no finite path can be a realisation of more than one finite execution in $\mathcal{M}^{\setminus sync}$ (since $\mathcal{M}^{\setminus sync}$ is deterministic). Let η be the set of all such finite executions where $\eta \subseteq Exec^{S_f}_{fin}(\mathcal{M}^{\setminus sync})$. From the definition of po-equivalence for MTSs (Def. 6.23) we know that for set η there exists a set $\eta' \subseteq Exec^{S'_f}_{fin}(\mathcal{M}'^{\setminus sync}) : \forall \rho \in \eta \exists \rho' \in \eta' : \rho \equiv^*_{po} \rho'$ and vice versa. Since both $\mathcal{M}^{\setminus sync}$ and $\mathcal{M}'^{\setminus sync}$ are deterministic, this intuitively means that $|\eta| = |\eta'|$. Let $\mathcal{T}' \in [\![\mathcal{M}'^{\setminus sync}]\!]$ be an LTS s.t. for every finite abstract execution $\rho' \in \eta'$ there exists a finite path in $\mathcal{T}'$ that is a realisation of ρ' and $\mathcal{T}'$ does not contain any path $\pi' \in Paths^{S'_f}_{fin}(\mathcal{T}')$ which is not a realisation of some finite execution $\rho' \in \eta'$. In other words we have constructed an implementation $\mathcal{T}'$ corresponding to $\mathcal{T}$ s.t. $\mathcal{T}' \models \Diamond S'_f$.

($\forall$ reachability): We provide the proof of $\mathcal{M} \models^{\forall} \Diamond S_f \Rightarrow \mathcal{M}' \models^{\forall} \Diamond S'_f$. The proof of $\mathcal{M}' \models^{\forall} \Diamond S'_f \Rightarrow \mathcal{M} \models^{\forall} \Diamond S_f$ is similar. From the definition of $\forall$ reachability (Def. 6.11) we know that $\mathcal{M}$ reaches S_f if and only if all the implementations of $\mathcal{M}$ are able to reach S_f. This intuitively means that $\mathcal{M}^{\setminus sync}$ does not have those *may* transitions that block any of its implementations from reaching the final state. Since $\mathcal{M}'$ is po-equivalent to $\mathcal{M}$ and $\mathcal{M}'$ involves the consecutive execution of actions of $\mathcal{M}_1$, followed by the consecutive execution of actions of $\mathcal{M}_2$ before reaching S'_f, such *may* transitions are also absent in $\mathcal{M}'^{\setminus sync}$. In other words, every implementation $\mathcal{T}' \in [\![\mathcal{M}'^{\setminus sync}]\!]$ reaches S'_f, i.e., $\mathcal{T}' \models \Diamond S'_f$. ■

Proof of Proposition 6.30

Proof: Let S_f, S'_f, S_{f1} be the set of final states in $\mathcal{M}, \mathcal{M}'$ and $\mathcal{M}_1$, respectively. We know that $\mathcal{M}$ and $\mathcal{M}'$ are po-equivalent, which means that for any $\rho \in Exec^{S_f}_{fin}(\mathcal{M}^{\setminus sync})$, there exists a $\rho' \in Exec^{S'_f}_{fin}(\mathcal{M}'^{\setminus sync})$ s.t. $\rho \equiv^*_{po} \rho'$ and vice versa. Now consider a finite execution ρ'' in $(\mathcal{M}||\mathcal{M}_1)^{\setminus sync}$ obtained by executing actions of ρ and ρ_1, where $\rho_1 \in Exec^{S_{f1}}_{fin}(\mathcal{M}_1^{\setminus sync})$. It is easy to show (similar to proof of Lemma 6.25, Thm. 6.27) that for ρ'', there exists a finite execution ρ''' in $(\mathcal{M}'||\mathcal{M}_1)^{\setminus sync}$ obtained by executing actions of ρ' and ρ_1 s.t. $\rho'' \equiv^*_{po} \rho'''$. ■

A.5 Proofs of Chapter 7

Proof of Theorem 7.17

Proof: We provide the proof of the law CCL-L. The proofs of the other CCL laws are similar. Let $\mathcal{T} = (\mathcal{N}_1 \bullet \mathcal{N}_2)||\mathcal{M}_2$ and $\mathcal{U} = \mathcal{N}_1 \bullet (\mathcal{N}_2||\mathcal{M}_2)$. In order to prove that $\mathcal{T} \equiv \mathcal{U}$, we have to show $\mathcal{T} \preceq_S \mathcal{U}$ and $\mathcal{U} \preceq_S \mathcal{T}$. Let $((x,y),z)$ and $(x,(y,z))$ denote states of APAs $\mathcal{T}$ and $\mathcal{U}$, respectively. Here x, y, z represents the state components of $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$, respectively. We only prove $\mathcal{T} \preceq_S \mathcal{U}$, the proof of $\mathcal{U} \preceq_S \mathcal{T}$ is very similar. To show $\mathcal{T} \preceq_S \mathcal{U}$ we have to prove that $\mathcal{T}$ strongly refines $\mathcal{U}$ according to Def. 7.7. Let S^* and S be the state space of $\mathcal{T}$ and $\mathcal{U}$. Let $\mathcal{R} \subseteq S^* \times S$ be a binary relation such that $\mathcal{R} = \{(((x,y),z),(x,(y,z))),(((x',y),z),(x',(y,z))),(((x,y'),z'),(x,(y',z')))\ldots\}$. Intuitively $\mathcal{R}$ relates states of S^* to those states in S that have the same individual state components from $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$. Let us consider a state $s = (x,(y,z))$ from $\mathcal{U}$ and $s^* = ((x,y),z)$ from $\mathcal{T}$. From the definition of $\mathcal{R}$ we know that $(s^*, s) \in \mathcal{R}$. A transition from s in $\mathcal{U}$ or s^* in $\mathcal{T}$ can be performed by 1) $\mathcal{N}_1$ individually, or 2) $\mathcal{N}_2$ individually, or 3) $\mathcal{M}_2$ individually, or 4) $\mathcal{N}_1$ and $\mathcal{N}_2$ simultaneously, or 5) $\mathcal{N}_2$ and $\mathcal{M}_2$ simultaneously. We show that for every case[1] the conditions of Def. 7.7 are satisfied, and thus $\mathcal{R}$ is a strong refinement relation between $\mathcal{T}$ and $\mathcal{U}$. It is easy to check that condition 3 which requires final states to be related is trivially satisfied. This is because $\mathcal{R}$ relates states that have the same individual state components from $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_2$.

1) $\mathcal{N}_1$ individually: Let $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) be a transition of $\mathcal{N}_1$ taken in $\mathcal{U}$. Since $\mathcal{N}_1$ is the left operand of layering in $\mathcal{U}$ such a transition is also possible from the related state $s^* = ((x,y),z)$, i.e., $s^* \xrightarrow{a}_\top \varphi'$ (resp. $s^* \xrightarrow{a}_? \varphi'$). Since this transition is similar to the transition from s, it is easy to check that $\forall \mu' \in Sat(\varphi') \exists \mu \in Sat(\varphi)$ s.t. $\mu \Subset_R^\delta \mu'$. For example, let $\mu'(s^{**}) = p, \mu'(s^{***}) = 1-p$ where $s^{**} = ((x',y),z), s^{***} = ((x'',y)z)$, and $\mu'(s^{'}) = p, \mu'(s^{''}) = 1-p$ where $s^{'} = (x',(y,z)), s^{''} = (x'',(y,z))$. Then we can do the assignment, $\delta(s^{**})(s^{'}) = 1$ and $\delta(s^{***})(s^{''}) = 1$. Since $\delta(s^{**})(s^{'}) > 0, \delta(s^{***})(s^{''}) > 0$, these states should be related, i.e., $(s^{**}, s') \in \mathcal{R}$ and $(s^{***}, s^{''}) \in \mathcal{R}$ (which is indeed the case from the definition of $\mathcal{R}$). Similarly it can be checked that for $s^* \xrightarrow{a}_\top \varphi'$ (resp. $s^* \xrightarrow{a}_? \varphi'$), there exists a corresponding transition, $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) s.t. the condition on relating distributions is satisfied.

[1] Note that we do not consider the case where $\mathcal{N}_1$ and $\mathcal{M}_2$ move simultaneously. This is due to the fact that $\mathcal{N}_1 \ddagger \mathcal{M}_2$, and therefore they cannot have common actions (since the dependency relation is reflexive).

2) $\mathcal{N}_2$ individually: Let $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) be a transition of $\mathcal{N}_2$ taken in $\mathcal{U}$. Since this transition is possible in $\mathcal{U}$, this intuitively means that action a is not waiting for actions from $\mathcal{N}_1$ to be executed. As $\mathcal{M}_2$ is in parallel composition it also does not respect dependencies. Thus a similar transition also exists in $\mathcal{T}$. Similarly, for every transition in $\mathcal{T}$, a similar transition exists in $\mathcal{U}$. The rest of the proof, showing that distributions are related is similar to case 1 above.
3) $\mathcal{M}_2$ individually: Let $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) be a transition of $\mathcal{M}_2$ taken in $\mathcal{U}$. Since this transition is possible after layering operator in $\mathcal{U}$, it is also possible in $\mathcal{T}$ as this action is not waiting for some action of $\mathcal{N}_1$ to be executed, and $\mathcal{U}$ induces fewer interleavings due to dominance of layered composition operator. Thus such a transition also exists in $\mathcal{T}$. Next, we consider a transition of $\mathcal{M}_2$ taken in $\mathcal{T}$. Since $\mathcal{N}_1 \ddagger \mathcal{M}_2$, a similar transition exists in $\mathcal{U}$, i.e., putting $\mathcal{M}_2$ after the layering operator is not a problem as it is independent from $\mathcal{N}_1$. The relation between distributions of the corresponding states can be shown similar to case 1.
4) $\mathcal{N}_1$ and $\mathcal{N}_2$ simultaneously: Let $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) be a transition in $\mathcal{U}$ as a result of a synchronization of $\mathcal{N}_1$ and $\mathcal{N}_2$ on action a. Again as $\mathcal{U}$ induces fewer interleavings due to dominance of layered operator, a similar transition is possible in $(\mathcal{N}_1 \bullet \mathcal{N}_2)$, i.e., $\mathcal{T}$. Similarly, for any transition in $\mathcal{T}$, a similar transition is enabled in $(\mathcal{N}_2||\mathcal{M}_2)$ (due to the fact that the synchronizing action is not waiting for an action from $\mathcal{M}_2$ to be executed since parallel composition does not respect dependencies). The relation between distributions of the corresponding states can be shown similar to case 1.
5) $\mathcal{N}_2$ and $\mathcal{M}_2$ simultaneously: Let $s \xrightarrow{a}_\top \varphi$ (resp. $s \xrightarrow{a}_? \varphi$) be a transition in $\mathcal{U}$ as a result of a synchronization of $\mathcal{N}_2$ and $\mathcal{M}_2$ on action a. As this action is possible in $(\mathcal{N}_2||\mathcal{M}_2)$, it means that this action is not waiting for the execution of some actions from $\mathcal{N}_1$. It is therefore possible to take the same transition in $\mathcal{T}$ as $(\mathcal{N}_1 \bullet \mathcal{N}_2)$ will not block it. Similarly, for any transition in $\mathcal{T}$, there will be a corresponding transition in $\mathcal{U}$. This is due to the fact this action is not waiting for actions of $\mathcal{N}_1$ to be executed and also we know that $\mathcal{N}_1 \ddagger \mathcal{M}_2$. The relation between distributions of the corresponding states can be shown similar to case 1. ■

Proof of Proposition 7.18
Proof: Let $\mathcal{N}$ be an APA, and $[\![\mathcal{N}]\!]$ be the set of all implementations of $\mathcal{N}$. We prove this proposition in two steps. In first step we show that to compute $P^{max}_{\mathcal{N}}(S_f)$ and $P^{min}_{\mathcal{N}}(S_f)$, it is sufficient to consider only a subset of implementations of $\mathcal{N}$ that are deterministic, i.e., $\mathcal{I} \subset [\![\mathcal{N}]\!]$ where $\forall \mathcal{P} \in \mathcal{I}$: $\mathcal{P}$ is deterministic. In the second step, we show that infact only those determin-

istic implementations are sufficient which satisfy the following condition: every finite path of the implementation is a realisation of some finite abstract execution of $\mathcal{N}$. More formally, $\mathcal{T} \subset \mathcal{I}$ where $\forall \mathcal{P} \in \mathcal{T} : \forall \pi \in Paths_{fin}(\mathcal{P}) \exists \rho \in Exec_{fin}(\mathcal{N}) : \pi \models \rho$. As mentioned in the paper we only consider deterministic APAs. Let $\mathcal{N}$ be a deterministic APA, i.e., no state in $\mathcal{N}$ can have two transitions on the same action. This intuitively means, that in an implementation $\mathcal{P} \in [\![\mathcal{N}]\!]$, if any state s has more than one outgoing transition on same action, then this is due to the single corresponding transition from the related state s' in APA $\mathcal{N}$. In other words, every distribution on the same action transition in $\mathcal{P}$ from state s is simulated by some distribution that satisfies the single constraint in the corresponding transition from s' (from the definition of satisfaction). Let $\mathcal{P} \in [\![\mathcal{N}]\!]$ s.t. only the initial state of $\mathcal{P}$, i.e., s_0 has n transitions on an action say a. In other words, $\mathcal{P}$ is nondeterministic only because s_0 has multiple transitions on action a. Let the distributions for these n transitions on action a are denoted by $\mu_1, \mu_2, \ldots, \mu_n$. This means that in state s_0 if some adversary[1] decides to take a action transition then it can choose one of these n transitions. Now consider n other implementations of $\mathcal{N}$ corresponding to $\mathcal{P}$ denoted by $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ where each implementation $\mathcal{P}_i$ (where $0 < i \leq n$) is obtained by taking a copy of $\mathcal{P}$ and removing all the a action transitions (and states that can be only reached by these transitions) from s_0 except the one with distribution μ_i. Note that these n implementations are deterministic (since $\mathcal{P}$ was nondeterministic only because of multiple a labeled transitions from s_0 and now in every implementation $\mathcal{P}_i$ we have only one transition on action a from s_0). Now if some adversary $\mathcal{D}$ in $\mathcal{P}$ takes a non-a transition from s_0, say action b then such a transition can be taken in every implementation from state s_0 by some adversary $\mathcal{D}_i \in Adv(\mathcal{P}_i)$ (since by construction we know that such a transition is possible in every $\mathcal{P}_i$). Since for this case any transition taken by $\mathcal{D}$ starting from initial state can also be taken in every $\mathcal{P}_i$ by some $\mathcal{D}_i$, $\mathcal{D}$ and all the corresponding $\mathcal{D}_i$ give the same reachability probability to reach the set of final states. On the other hand, if an adversary $\mathcal{D}$ in $\mathcal{P}$ takes a transition from s_0 labeled with action a and distribution μ_i, then such a transition can also be taken in a specific implementation, i.e., $\mathcal{P}_i$ by some adversary $\mathcal{D}_i$ from s_0 s.t. reachability probabilities to reach the set of final states coincide. Again, the reachability probabilities coincide as both $\mathcal{D}$ in $\mathcal{P}$ and $\mathcal{D}_i$ in $\mathcal{P}_i$ take the same actions with same probabilities starting from initial states. Thus, we can safely remove $\mathcal{P}$ from the set of implementations of

[1] As mentioned earlier, we only consider history-independent adversaries, and therefore the next action is chosen based on the current state.

$[\![\mathcal{N}]\!]$, and only consider the n corresponding implementations for computing the extremal reachability probabilities in $\mathcal{P}$.

Now, if for some $\mathcal{P} \in [\![\mathcal{N}]\!]$, s_0 has m transitions on action a and n transitions on action b and all other states are deterministic, then we can consider $m \times n$ other implementations of $\mathcal{N}$ corresponding to $\mathcal{P}$ where each implementation, i.e., $\mathcal{P}_{i,j}$ (where $0 < i \leq m$, $0 < j \leq n$) is obtained by taking a copy of $\mathcal{P}$ and removing all the a and b action transitions (and staes that can be only reached by these transitions) from s_0 except one a transition and one b transition with distributions μ_i and μ_j, respectively. Again it can be observed that all these implementations are deterministic and $P^{max}_{\mathcal{P}}(S_f) = sup_{0<i\leq m, 0<j\leq n} P^{max}_{\mathcal{P}_{ij}}(S_f)$. The same holds for minimum reachability properties. Thus we can safely remove $\mathcal{P}$ from implementations of $\mathcal{N}$ and consider the $m \times n$ deterministic implementations in place of $\mathcal{P}$ for computing the $P^{max}_{\mathcal{N}}(S_f)$ and $P^{min}_{\mathcal{N}}(S_f)$. Similarly, for every arbitrary nondeterministic implementation of $\mathcal{N}$ (where any state can have multiple transitions on every action that is enabled), we have a set of corresponding deterministic implementations of $\mathcal{N}$ s.t. if we remove the nondeterministic implementation from $[\![\mathcal{N}]\!]$ then we still get the same values for $P^{max}_{\mathcal{N}}(S_f)$ and $P^{min}_{\mathcal{N}}(S_f)$. Thus we can safely remove all the nondeterministic implementations from $[\![\mathcal{N}]\!]$. Let $\mathcal{I} \subset [\![\mathcal{N}]\!]$ be the set of all deterministic implementations of $\mathcal{N}$ that are sufficient for computing extremal reachability properties.

Next, we show that we do not need to consider all the deterministic implementations in $\mathcal{I}$ for computing $P^{max}_{\mathcal{N}}(S_f)$ and $P^{min}_{\mathcal{N}}(S_f)$, and it is sufficient to consider only a subset $\mathcal{T} \subset \mathcal{I}$ that satisfy the condition that $\forall \mathcal{P} \in \mathcal{T} : \forall \pi \in Paths_{fin}(\mathcal{P}) \exists \rho \in Exec_{fin}(\mathcal{N}) : \pi \models \rho$. Let $\mathcal{P} \in \mathcal{I}$ that satisfies this condition. We know that every PA is an APA. If we consider $\mathcal{P}$ to be an APA, then we can relate many PAs in $\mathcal{I}$ with $\mathcal{P}$ via satisfaction relation (Def. 7.2). It is easy to see that all the implementations in $\mathcal{I}$ that can be related with $\mathcal{P}$ are deterministic (since $\mathcal{I}$ is set of deterministic implementations). Since in $\mathcal{P}$ every transition is a must transition, any action that can be taken from state s in $\mathcal{P}$, can also be taken from all the related states in every implementation of $\mathcal{P}$. In other words, if two states s, s' are related by the satisfaction relation, then $act(s) = act(s')$. From the definition of satisfaction we know that the initial states of all these implementations agree on the probability distributions with the initial state of $\mathcal{P}$. This means that if an adversary $\mathcal{D}$ from the initial state of $\mathcal{P}$ takes the a labeled transition with distribution μ, then in every implementation of $\mathcal{P}$, say $\mathcal{P}_i \in \mathcal{I}$, some $\mathcal{D}_i$ can also take the a labeled transition from the intial state with some distribution μ' s.t. $\mu' \Subset_R \mu$, where R is the satisfaction relation. Since

every transition taken by $\mathcal{D}$ in $\mathcal{P}$ with distribution μ is matched by a corresponding transition taken by $\mathcal{D}_i$ in $\mathcal{P}_i$ with distribution μ' s.t. $\mu' \Subset_R \mu$, thus they all have the same maximum (resp. minimum) probability to reach set of final states. For this reason, it is sufficient to just consider $\mathcal{P}$ and remove all the implementations of $\mathcal{P}$ from $\mathcal{I}$ (that are related by satisfaction relation). Similarly, all the other deterministic implementations that do not satisfy the condition that $\forall \pi \in Paths_{fin}(\mathcal{P}) \exists \rho \in Exec_{fin}(\mathcal{N}) : \pi \models \rho$ can be removed from $\mathcal{I}$.

■

Proof of Theorem 7.19

Proof: In order to prove this theorem we show that for every $\mathcal{P} \in [\![\mathcal{N}_1 \bullet \mathcal{N}_2]\!]$ and every $\mathcal{D} \in Adv(\mathcal{P})$, we can construct a corresponding $\mathcal{P}' \in [\![(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}]\!]$ and $\mathcal{D}' \in Adv(\mathcal{P}')$ s.t. for every finite path π under $\mathcal{D}$ where $last(\pi) = s'$, there is a finite path π' (where $last(\pi') = s'$) without synchronized edges in $\mathcal{D}'$ with the same probability. Let $\mathcal{N} = \mathcal{N}_1 \bullet \mathcal{N}_2$, $\mathcal{P} \in [\![\mathcal{N}]\!]$, $\mathcal{D} \in Adv(\mathcal{P})$ and $\Pi \subseteq Paths_{fin}(\mathcal{P})$ s.t. $\pi \in \Pi$ iff $\pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$. We know that every path $\pi \in Paths_{fin}(\mathcal{P})$ is a realisation of some finite abstract execution $\rho \in Exec_{fin}(\mathcal{N})$ and no two finite paths of $\mathcal{P}$ can be the realisation of same $\rho \in Exec_{fin}(\mathcal{N})$ (by Proposition 7.18). We also know that for all such finite abstract executions, say set η, there are corresponding finite abstract executions in $\mathcal{N}^{\setminus sync}$, say set η', obtained by allowing interleaving on common actions s.t. the common action of $\mathcal{N}_1$ is executed first followed by the common action of $\mathcal{N}_2$. Let $\eta* \subseteq \eta$ s.t. for every abstract execution $\rho \in \eta*$, there is a path $\pi \in \Pi$ that is a realisation of ρ, and every path $\pi \in \Pi$ is a realisation of some $\rho \in \eta*$. Consider $\mathcal{P}'$ to be an implementation of $\mathcal{N}^{\setminus sync}$ s.t. for every finite abstract execution $\rho' \in \eta'$ there exists a path in $\mathcal{P}'$ that is a realisation of ρ' and $\mathcal{P}'$ does not contain any path π which is not a realisation of some $\rho' \in \eta'$. From proposition 7.18 we also know that no two paths of $\mathcal{P}'$ can be the realisation of same $\rho' \in Exec_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ (since we only consider realisable implementations). For this $\mathcal{P}'$, let $\mathcal{D}' \in Adv(\mathcal{P}')$, and let $\eta'' \subseteq \eta'$ where for every finite abstract execution $\rho' \in \eta''$ there is a path $\pi' \in Paths^{\mathcal{D}'}_{fin}(\mathcal{P}')$ with $\pi' \models \rho'$ and every path $\pi' \in Paths^{\mathcal{D}'}_{fin}(\mathcal{P}')$ is a realisation of some $\rho' \in \eta''$ (this is possible as we only consider realisable implementations, from Proposition 7.18). We choose this adversary $\mathcal{D}' \in Adv(\mathcal{P}')$ such that for every finite path π' (under $\mathcal{D}'$) we have $\pi' \models \rho'$ for some $\rho' \in \eta''$, and for every $\rho \in \eta*$, there is a corresponding finite abstract execution without synchronized transitions in the set η''. Note that from the definition of parallel composition it is known that for any synchronized transition in a finite abstract execution, the new constraint $\tilde{\varphi}$ is s.t. $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff there exists $\mu \in Sat(\varphi)$ and $\mu' \in Sat(\varphi')$ such that $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$ for

all $u \in S_1, v \in S_2$.

So far we have constructed an implementation $\mathcal{P}'$ and adversary $\mathcal{D}'$, but it can be clearly observed that there can be multiple implementations of $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}$ that satisfy the condition that every finite abstract execution $\rho' \in \eta'$ there exists a path in $\mathcal{P}'$ that is a realisation of ρ' and $\mathcal{P}'$ does not contain any path π which is not a realisation of some $\rho' \in \eta'$. We therefore choose $\mathcal{P}'$ from these implementations s.t. 1) for every non-synchronized transitions of type $s \xrightarrow{a}_\top \varphi$ or $s \xrightarrow{a}_? \varphi$, if $\mu \in Sat(\varphi)$ is a distribution in $\mathcal{P}$ then $\mu \in Sat(\varphi)$ is also a distribution in $\mathcal{P}'$ for every corresponding transition of type $s \xrightarrow{a}_\top \varphi$ or $s \xrightarrow{a}_? \varphi$. 2) similarly, for every synchronized transition, if $\tilde{\mu} \in Sat(\tilde{\varphi})$ is a distribution in $\mathcal{P}$, then for the corresponding sequence of transitions where common action of $\mathcal{N}_1$ is executed first followed by common action of $\mathcal{N}_2$, we have distributions $\mu \in Sat(\varphi), \mu' \in Sat(\varphi')$ in $\mathcal{P}'$ where $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$. These conditions make sure that for every finite path π in $\mathcal{P}$ under D, there exists a corresponding finite path π' in P' under D' with the same probability. More formally, $\mathcal{P}'$ and $\mathcal{D}'$ have been constructed s.t., $\forall \mathcal{P} \in [\![\mathcal{N}]\!] \forall D \in Adv(\mathcal{P}) \forall \pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P}) : \exists \mathcal{P}' \in [\![\mathcal{N}^{\setminus sync}]\!] \exists D' \in Adv(\mathcal{P}') \exists \pi' \in Paths^{\mathcal{D}'}_{fin}(\mathcal{P}') : P^{\mathcal{D}}(\pi) = P^{\mathcal{D}'}(\pi')$. ■

Example A.1 (Intuition behind Thm. 7.19) *Consider the APA $\mathcal{N}_1$ and $\mathcal{N}_2$ (left) shown in Fig. A.2. The layered composition of $\mathcal{N}_1, \mathcal{N}_2$, i.e., $\mathcal{N}_1 \bullet \mathcal{N}_2$ is shown in the middle, where synchronization takes place on common action a. A may transition is obtained in $\mathcal{N}_1 \bullet \mathcal{N}_2$ (since synchronization of must-may results in a may transition). Now consider the APA $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}$ where we allow interleaving on common actions s.t. the common action of $\mathcal{N}_1$ is executed before the common action of $\mathcal{N}_2$. It is easy to check that for every implementation of $\mathcal{N}_1 \bullet \mathcal{N}_2$ (that satisfies the conditions of Proposition 7.18) and adversary $\mathcal{D}$, we can construct a corresponding implementation of $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}$ (that satisfies the conditions of Proposition 7.18) and adversary $\mathcal{D}'$ s.t. every path under $\mathcal{D}$ is matched by a path under $\mathcal{D}'$ with the same probability. Note that if we did not restrict an APA to be deterministic, $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync}$ would yield the automaton shown in the Fig. A.2 (extreme right).*

Proof of Lemma 7.22

Proof: From the definition of layered normal form (LNF) we know that all finite abstract executions that are in LNF consists of the consecutive execution of actions of $\mathcal{N}_1$, followed by the consecutive execution of actions of $\mathcal{N}_2$. We know that in $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$, an action of $\mathcal{N}_2$ occurs only when all the actions in $\mathcal{N}_1$

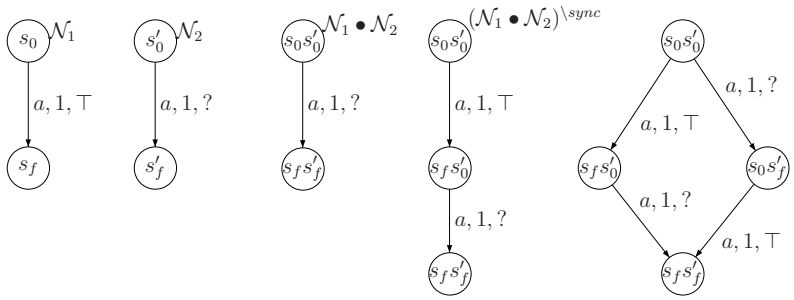

Figure A.2: APA without synchronized transitions

on which it is dependent have been executed. This intuitively means that all the actions of $\mathcal{N}_2$ that occur in a finite abstract execution before any action of $\mathcal{N}_1$ are independent of this action and thus by repeated permutation of these actions any finite abstract execution $\rho \in Exec^{S_f}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ can be converted to a finite abstract execution that is in LNF. ■

Proof of Theorem 7.24

Proof: Let $\mathcal{N}_1, \mathcal{N}_2$ be APAs. From the definition of LNF (Def. 7.21) we know that a finite abstract execution in LNF involves the consecutive execution of actions of $\mathcal{N}_1$, followed by the consecutive execution of actions of $\mathcal{N}_2$. This means that for every finite abstract execution $\rho \in Exec^{LNF}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ there exists a finite abstract execution ρ' in $((\mathcal{N}_1; \mathcal{N}_2)^{\setminus sync})$ that ends in the final state and where: $\forall n \geq 0 : \rho[n] \approx \rho'[n] \wedge \rho[n] \xrightarrow{a_{n+1}}_{\top} \varphi_{n+1} \Rightarrow \rho'[n] \xrightarrow{a_{n+1}}_{\top} \varphi_{n+1}$(resp. $\rho[n] \xrightarrow{a_{n+1}}_{?} \varphi_{n+1} \Rightarrow \rho'[n] \xrightarrow{a_{n+1}}_{?} \varphi_{n+1}) \wedge \forall \mu \in Sat(\varphi_{n+1}) : \mu(\rho[n+1]) = \mu(\rho'[n+1])$. The relation $\approx$ between states of $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ and $((\mathcal{N}_1; \mathcal{N}_2)^{\setminus sync})$ is defined as follows: $S_1 \times S_2$ is the state space of $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ and $(S_1 \setminus \{s_{f1}\} \cup S_2)$ is the state space of $((\mathcal{N}_1; \mathcal{N}_2)^{\setminus sync})$ then $\forall s_1 \in S_1$ where $s_1 \neq s_{f1} : (s_1, s_{02}) \approx s_1$ and $\forall s_2 \in S_2 : (s_{f1}, s_2) \approx s_2$.

In other words we have related every finite abstract execution of $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ that is in LNF to some finite abstract execution in $((\mathcal{N}_1; \mathcal{N}_2)^{\setminus sync})$ that ends in the final state and vice-versa. It is also clear from Lemma 7.22 that for every finite abstract execution ρ in $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ that ends in the final state there exists a finite abstract execution ρ' in $((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\setminus sync})$ that is LNF s.t. $\rho \equiv^*_{po} \rho'$. Hence proved. ■

Proof of Proposition 7.25

Proof: The proof of this proposition is similar to the proof of Proposition 6.30.

■

Proof of Theorem 7.26

Proof: Let $\mathcal{P} \in [\![\mathcal{N}_1^{\setminus sync}]\!]$ be an implementation and $\mathcal{D} \in Adv(\mathcal{P})$ s.t. the probability to reach the set of final states in $\mathcal{P}$ under $\mathcal{D}$ is maximum over all implementations. We know that every finite path of $\mathcal{P}$ that ends in the final state, i.e., S_{f1}, is a realisation of some finite abstract execution $\rho \in Exec_{fin}^{S_{f1}}(\mathcal{N}_1^{\setminus sync})$ and no two paths of $\mathcal{P}$ that ends in the final state can be a realisation of same $\rho \in Exec_{fin}^{S_{f1}}(\mathcal{N}_1^{\setminus sync})$ (proposition 7.18). Let η be the set of all such finite abstract executions. Since $\mathcal{N}_1 \equiv_{po}^{*} \mathcal{N}_2$, we know that there is a set η' of finite abstract executions in $\mathcal{N}_2^{\setminus sync}$ s.t. $\forall \rho \in \eta\ \exists \rho' \in \eta' : \rho \equiv_{po}^{*} \rho'$ and vice versa. Let $\mathcal{P}'$ be an implementation of $\mathcal{N}_2^{\setminus sync}$, i.e., $\mathcal{P}' \in [\![\mathcal{N}_2^{\setminus sync}]\!]$ s.t. for every $\rho' \in \eta'$ there is a path $\pi' \in Paths_{fin}^{S_{f2}}(\mathcal{P}')$ which is a realisation of ρ' and $\mathcal{P}'$ does not contain any path that ends in the final state and is not a realisation of some finite abstract execution $\rho' \in \eta'$. From Proposition 7.18 we also know that no two paths in $\mathcal{P}'$ that ends in the final state can be the realisation of same $\rho' \in Exec_{fin}^{S_{f2}}(\mathcal{N}_2^{\setminus sync})$. Let $\eta* \subseteq \eta$ be the set of finite abstract executions s.t. for every $\rho \in \eta*$, there is a path $\pi \in Paths_{fin}^{\mathcal{D},S_{f1}}(\mathcal{P})$ which is a realisation of $\eta*$, and every path in $Paths_{fin}^{\mathcal{D},S_{f1}}(\mathcal{P})$ should be a realisation of some finite abstract execution $\rho \in \eta*$. Similarly we construct $\eta'' \subseteq \eta'$ s.t. for every $\rho' \in \eta''$ there is path $\pi' \in Paths_{fin}^{\mathcal{D}',S_{f2}}(\mathcal{P}')$ which is a realisation of ρ' and every path $\pi' \in Paths_{fin}^{\mathcal{D}',S_{f2}}(\mathcal{P}')$ should be a realisation of some $\rho' \in \eta''$. We choose this adversary $\mathcal{D}' \in Adv(\mathcal{P}')$ such that for every finite path π' (under $\mathcal{D}'$) that ends in the final state, we have $\pi' \models \rho'$ for some $\rho' \in \eta''$, and $\forall \rho \in \eta * \exists \rho' \in \eta'' : \rho \equiv_{po}^{*} \rho'$ and vice versa. So far we have constructed an implementation $\mathcal{P}'$ and adversary $\mathcal{D}'$, but there can be multiple implementations of $\mathcal{N}_2^{\setminus sync}$ that satisfy the condition that every finite abstract execution $\rho' \in \eta'$ there exists a path in $\mathcal{P}'$ that is a realisation of ρ' and $\mathcal{P}'$ does not contain any path π that ends in the final state and is not a realisation of some $\rho' \in \eta'$. We therefore need to choose an implementation $\mathcal{P}'$ (which is constructed according to the above mentioned steps) and for which $\mathcal{D}'$ gives the maximum probability to reach set of final states (and exactly the same value that $\mathcal{D}$ provides for $\mathcal{P}$). We choose $\mathcal{P}'$ s.t. for every transition of type $s \xrightarrow{a}_{\top} \varphi$ or $s \xrightarrow{a}_{?} \varphi$, if $\mu \in Sat(\varphi)$ is a distribution in $\mathcal{P}$ under $\mathcal{D}$, then $\mu \in Sat(\varphi)$ is also a distribution for corresponding transition of type $s' \xrightarrow{a}_{\top} \varphi$ or $s' \xrightarrow{a}_{?} \varphi$ in $\mathcal{P}'$ under $\mathcal{D}'$. For all other transitions, any distributions that satisfies the corresponding constraints (according to satisfaction relation) can be choosen. If we choose $\mathcal{P}'$ according to the above mentioned condition then for every path $\pi = \pi^* a_1 \mu_1 s_1 a_2 \mu_2 \pi^{**}$ in $Paths_{fin}^{\mathcal{D},S_{f1}}(\mathcal{P})$ we either

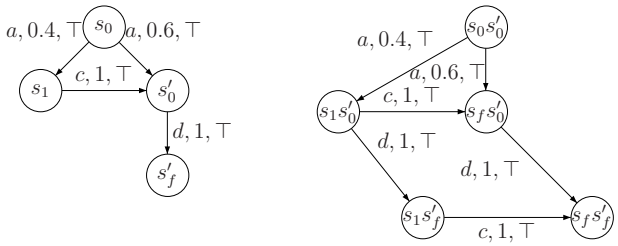

Figure A.3: Property preservation under po-equivalence for APAs

have a path $\pi' = \pi^* a_2 \mu_2 s'_1 a_1 \mu_1 \pi^{**}$ or $\pi'' = \pi^* a_1 \mu_1 s'_1 a_2 \mu_2 \pi^{**}$ in $Paths_{fin}^{\mathcal{D}', S_{f2}}(\mathcal{P}')$ s.t. π' can be obtained from π by repeated permutation of adjacent independent actions and $P^{\mathcal{D}}(\pi) = P^{\mathcal{D}'}(\pi')$. A similar proof can be constructed showing the preservation of minimum reachability probabilities computed over all the implementations. ■

Example A.2 (Intuition behind Thm. 7.26) *Consider the APA $\mathcal{N}_1;\mathcal{N}_2$ in Fig. 7.2 (right) and APA $\mathcal{N}_1 \bullet \mathcal{N}_2$ in Fig. 7.3 (right). From Thm. 7.24 we know that $\mathcal{N}_1;\mathcal{N}_2 \equiv_{po}^{*} \mathcal{N}_1 \bullet \mathcal{N}_2$. Now let us consider an implementation of $\mathcal{N}_1 \bullet \mathcal{N}_2$, say $\mathcal{P}$ shown in the Fig. A.3 (right). According to our construction in proof of Thm. 7.26 we can always construct a corresponding implementation of $\mathcal{N}_1;\mathcal{N}_2$, say $\mathcal{P}'$ shown in Fig. A.3 (left). From the adversary construction in proof of Thm. 7.26 for any adversary $\mathcal{D} \in Adv(\mathcal{P})$ we can construct a corresponding adversary $\mathcal{D}' \in Adv(\mathcal{P}')$ s.t. for every path π under $\mathcal{D}$ that ends in the final state there is a corresponding path π' under $\mathcal{D}'$ from which π can be obtained by repeated permutation of independent actions. From Fig. A.3, it is easy to check that this is indeed the case. For example, if on the right hand side adversary chooses c action from state $s_1 s'_0$ then for all paths in this system we have corresponding paths in the system on the left. On the other hand if some other adversary chooses d action from state $s_1 s'_0$, then for path that executes action a, d and then c we have a corresponding path in system on left where actions d and c can be permuted to get the same sequence. Note that probability of paths is preserved.*

A.6 Proofs of Chapter 8

Proof of Theorem 8.9

Proof: The proof of this theorem is similar to proof of Thm. 5.9 and proof of Thm. 3.8. ■

Proof of Theorem 8.12

Proof: Let $s_1 \sim s_2$. We show that all the conditions of IME are satisfied, i.e., $\sim \Rightarrow IME$. The labeling condition is trivially satisfied. Let $s_1, s_2 \in C$ then there can be following three cases:

- $C \in I(S)$: the proof for this case is similar to proof of Thm. 3.12.
- $C \in M(S)$: the proof of this case is similar to proof of Lemma 5.12.
- $C \notin M(S) \wedge C \notin I(S)$: Since the last two conditions of Def. 8.11 coincide with the fourth condition of Def. 8.4, it is easy to see that C is also an equivalence class under IME.

Since all the conditions of IME are satisfied, we can conclude that $\sim \Rightarrow IME$. From Fig. 8.1 (left) it is easy to see that states that are not bisimilar can still be merged in one equivalence class under IME. Thus $\sim$ is strictly finer than IME. ■

Proof of Theorem 8.21

Proof: The proof of this theorem is similar to proof of Thm. 5.9 and proof of Thm. 3.8. ■

Proof of Theorem 8.25

Proof: Let $s_1 \approx s_2$. We show that all the conditions of WIME are satisfied, i.e., $\approx \Rightarrow WIME$. The labeling condition is trivially satisfied. Let $s_1, s_2 \in C$ then there can be following three cases:

- $C \in I(S)$: the proof for this case is similar to proof of Thm. 3.33.
- $C \in M(S)$: the proof of this case is similar to proof of Lemma 5.12.
- $C \notin M(S) \wedge C \notin I(S)$: Since the last two conditions of Def. 8.23 coincide with the fourth condition of Def. 8.15, it is easy to see that C is also an equivalence class under IME.

Since all the conditions of WIME are satisfied, we can conclude that $\approx \Rightarrow WIME$. From Fig. 8.2 (left) it is easy to see that states that are not weak bisimilar can still be merged in one equivalence class under WIME. Thus $\approx$ is strictly finer than WIME. ■

Proof of Theorem 8.26

Proof: Let s_1, s_2 are IME related. We show that s_1, s_2 are also WIME related, i.e., $IME \Rightarrow WIME$.

- $L(s_1) = L(s_2)$ from definition of IME.
- Let $s_1, s_2 \in C$ and $C \in M(S)$. Then $\forall D \in S/_{\mathcal{R}}$, $s', s'' \in Pred(C)$ we have $wr(s', C, , D) = wr(s'', C, D)$ (from definition of IME).
- Let $s_1, s_2 \in C$ and $C \in I(S)$. Then $\forall D \in S/_{\mathcal{R}}, \forall s', s'' \in Pred(C)$ we have $Pbr(s', C, D) = Pbr(s'', C, D)$. This condition also satisfies condition 2 of Def. 8.15 as condition 2 only requires that for $s', s'' \notin C, C \neq D$, two (or more) step reachability is satisfied.
- Let $s_1, s_2 \in C$ and $C \notin I(S) \wedge C \notin M(S)$. Then $\forall s_1, s_2 \in C : R(s_1, D) = R(s_2, D)$ for any $D \in S/_{\mathcal{R}}$. This condition also satisfies the subcondition 2 of condition 4 (Def. 8.15) as subcondition 2 only requires that if s_1 reaches s' in zero or more steps then s_2 should also reach s'' in zero or more steps s.t. $R(s_1, D) = R(s_2, D)$ for any $D \neq C$. Similarly subcondition 2 of condition 4 (Def. 8.4) also satisfies subcondition 1 of condition 4 (Def. 8.15) and therefore we can conclude that $IME \Rightarrow WIME$.

From Fig. 8.2 (left) it is easy to check that states s_5, s_6, s_7 cannot be merged under IME. Thus IME is strictly finer than WIME. ■

Bibliography

[1] Rajeev Alur and David L. Dill. "A Theory of Timed Automata". In: *Theor. Comput. Sci.* 126.2 (1994), pp. 183–235.

[2] Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. "Discrete-Time Rewards Model-Checked". In: *FORMATS*. LNCS 2791. Springer, 2003, pp. 88–104.

[3] Robert B. Ash and Catherine A. Doléans-Dade. *Probability and Measure Theory*. Academic Press, 2000.

[4] Adnan Aziz, Vigyan Singhal, and Felice Balarin. "It Usually Works: The Temporal Logic of Stochastic Systems". In: *CAV*. LNCS 939. Springer, 1995, pp. 155–165.

[5] Adnan Aziz, Vigyan Singhal, Gitanjali Swamy, and Robert K. Brayton. "Minimizing Interacting Finite State Machines: A Compositional Approach to Language Containment". In: *ICCD*. 1994, pp. 255–261.

[6] Jos C. M. Baeten, Jan A. Bergstra, and Jan W. Klop. "Ready-Trace Semantics for Concrete Process Algebra with the Priority Operator". In: *Comput. J.* 30.6 (1987), pp. 498–506.

[7] Christel Baier, Pedro R. D'Argenio, and Marcus Größer. "Partial Order Reduction for Probabilistic Branching Time". In: *Electr. Notes Theor. Comput. Sci.* 153.2 (2006), pp. 97–116.

[8] Christel Baier, Marcus Größer, and Frank Ciesinski. "Partial Order Reduction for Probabilistic Systems". In: *QEST*. IEEE, 2004, pp. 230–239.

[9] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[10] Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. "Comparative branching-time semantics for Markov chains". In: *Inf. Comput.* 200.2 (2005), pp. 149–214.

[11] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. "Model-checking algorithms for continuous-time Markov chains". In: *IEEE Trans. Software Eng.* 29.6 (2003), pp. 524–541.

[12] Benoît Barbot, Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. "Efficient CTMC model checking of linear real-time objectives". In: *TACAS*. LNCS 6605. Springer, 2011, pp. 128–142.

[13] Sebastian S. Bauer, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. "A Modal Specification Theory for Components with Data". In: *FACS*. LNCS 7253. Springer, 2011, pp. 61–78.

[14] Nikola Benes, Ivana Cerná, and Jan Kretínský. "Modal Transition Systems: Composition and LTL Model Checking". In: *ATVA*. LNCS 6996. Springer, 2011, pp. 228–242.

[15] Jan A. Bergstra. *Handbook of Process Algebra*. Elsevier Science Inc., 2001.

[16] Marco Bernardo. "Non-bisimulation-based Markovian behavioral equivalences". In: *J. Log. Algebr. Program.* 72.1 (2007), pp. 3–49.

[17] Marco Bernardo. "Towards State Space Reduction Based on T-Lumpability-Consistent Relations". In: *EPEW*. LNCS 5261. Springer, 2008, pp. 64–78.

[18] Marco Bernardo. "Uniform Logical Characterizations of Testing Equivalences for Nondeterministic, Probabilistic and Markovian Processes". In: *ENTCS* 253.3 (2009), pp. 3–23.

[19] Marco Bernardo and Stefania Botta. "A survey of modal logics characterising behavioural equivalences for non-deterministic and stochastic systems". In: *Math. Structures in Comp. Sci.* 18.1 (2008), pp. 29–55.

[20] Philip A Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1986.

[21] Eckard Böde, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Jan Rakow, Ralf Wimmer, and Bernd Becker. "Compositional Dependability Evaluation for STATEMATE". In: *IEEE Trans. Software Eng.* 35.2 (2009), pp. 274–292.

[22] Henrik C. Bohnenkamp, Peter van der Stok, Holger Hermanns, and Frits W. Vaandrager. "Cost-Optimization of the IPv4 Zeroconf Protocol". In: *DSN*. 2003, pp. 531–540.

[23] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. "A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis". In: *IEEE Trans. Dependable Sec. Comput.* 7.2 (2010), pp. 128–143.

[24] Patricia Bouyer. "From Qualitative to Quantitative Analysis of Timed Systems". Mémoire d'habilitation. Université Paris 7, France, Jan. 2009.

[25] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. "Safety, Dependability and Performance Analysis of Extended AADL Models". In: *Comput. J.* 54.5 (2011), pp. 754–775.

[26] Stephen D. Brookes, Charles A. R. Hoare, and Andrew W. Roscoe. "A Theory of Communicating Sequential Processes". In: *J. ACM* 31.3 (1984), pp. 560–599.

[27] Michael C. Browne, Edmund M. Clarke, and Orna Grumberg. "Characterizing Finite Kripke Structures in Propositional Temporal Logic". In: *Theor. Comput. Sci.* 59 (1988), pp. 115–131.

[28] Glenn Bruns. "An industrial application of modal process logic". In: *Sci. Comput. Program.* 29.1-2 (July 1997), pp. 3–22.

[29] Peter Buchholz. "Exact and ordinary lumpability in finite Markov chains". In: *J. of Appl$\dot{P}$rob.* (1994), pp. 59–75.

[30] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. "Constraint Markov Chains". In: *Theor. Comput. Sci.* 412.34 (2011), pp. 4373–4404.

[31] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. "Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications". In: *LICS*. IEEE, 2009, pp. 309–318.

[32] Ling Cheung, Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. "Switched PIOA: Parallel composition via distributed scheduling". In: *Theor. Comput. Sci.* 365.1-2 (2006), pp. 83–108.

[33] Ivan Christoff. "Testing Equivalences and Fully Abstract Models for Probabilistic Processes". In: *CONCUR*. LNCS 458. Springer, 1990, pp. 126–140.

[34] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. "NUSMV: A New Symbolic Model Checker". In: *STTT* 2.4 (2000), pp. 410–425.

[35] Edmund M. Clarke and Ernest A. Emerson. “Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic”. In: *Logic of Programs*. 1981, pp. 52–71.

[36] Edmund M. Clarke, Orna Grumberg, and David E. Long. “Model Checking and Abstraction”. In: *ACM Trans. Program. Lang. Syst.* 16.5 (1994), pp. 1512–1542.

[37] Rance Cleaveland, Scott A. Smolka, and Amy E. Zwarico. “Testing Preorders for Probabilistic Processes”. In: *ICALP*. LNCS 623. Springer, 1992, pp. 708–719.

[38] Nicolas Coste, Hubert Garavel, Holger Hermanns, Richard Hersemeule, Yvain Thonnart, and Meriem Zidouni. “Quantitative Evaluation in Embedded System Design: Validation of Multiprocessor Multithreaded Architectures”. In: *DATE*. 2008, pp. 88–89.

[39] Nicolas Coste, Holger Hermanns, Etienne Lantreibecq, and Wendelin Serwe. “Towards Performance Prediction of Compositional Models in Industrial GALS Designs”. In: *CAV*. LNCS 5643. Springer, 2009, pp. 204–218.

[40] Patrick Cousot and Radhia Cousot. “On Abstraction in Software Verification”. In: *CAV*. LNCS 2404. Springer, 2002, pp. 37–56.

[41] Christian Dax, Felix Klaedtke, and Stefan Leue. “Specification Languages for Stutter-Invariant Regular Properties”. In: *ATVA*. LNCS 5799. Springer, 2009, pp. 244–254.

[42] Rocco De Nicola and Frits W. Vaandrager. “Action versus State based Logics for Transition Systems”. In: *Semantics of Systems of Concurrent Processes*. 1990, pp. 407–419.

[43] Rocco De Nicola, Alessandro Fantechi, Stefania Gnesi, and Gioia Ristori. “An Action Based Framework for Verifying Logical and Behavioural Properties of Concurrent Systems”. In: *CAV*. LNCS 575. Springer, 1991, pp. 37–47.

[44] Benoît Delahaye, Joost-Pieter Katoen, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, Falak Sher, and Andrzej Wasowski. “Abstract probabilistic automata”. In: *VMCAI*. LNCS 6538. Springer, 2011, pp. 324–339.

[45] Benoît Delahaye, Joost-Pieter Katoen, Kim G. Larsen, Axel Legay, Mikkel Pedersen, Falak Sher, and Andrzej Wasowski. "Abstract Probabilistic Automata". In: *Inf. Comput.* 232 (2013), pp. 66–116.

[46] Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. "APAC: A Tool for Reasoning about Abstract Probabilistic Automata". In: *QEST*. IEEE, 2011, pp. 151–152.

[47] Benoît Delahaye, Joost-Pieter Katoen, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, Falak Sher, and Andrzej Wasowski. "New Results on Abstract Probabilistic Automata". In: *ACSD*. IEEE, 2011, pp. 118–127.

[48] Salem Derisavi, Holger Hermanns, and William H. Sanders. "Optimal state-space lumping in Markov chains". In: *Inf. Process. Lett.* 87.6 (2003), pp. 309–315.

[49] Josee Desharnais, Abbas Edalat, and Prakash Panangaden. "A Logical Characterization of Bisimulation for Labeled Markov Processes". In: *LICS*. IEEE, 1998, pp. 478–487.

[50] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. "Model Checking Timed and Stochastic Properties with CSL^{TA}". In: *IEEE Trans. Software Eng.* 35.2 (2009), pp. 224–240.

[51] Tzilla Elrad and Nissim Francez. "Decomposition of Distributed Programs into Communication-Closed Layers". In: *Sci. Comput. Program.* 2.3 (1982), pp. 155–173.

[52] Ernest A. Emerson and Thomas Wahl. "Dynamic Symmetry Reduction". In: *TACAS*. Vol. LNCS 3440. Springer, 2005, pp. 382–396.

[53] Harald Fecher, Martin Leucker, and Verena Wolf. "*Don't Know* in Probabilistic Systems". In: *SPIN*. LNCS 3925. Springer, 2006, pp. 71–88.

[54] William Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley and Sons, 2001.

[55] Kathi Fisler and Moshe Y. Vardi. "Bisimulation Minimization in an Automata-Theoretic Verification Framework". In: *FMCAD*. LNCS 1522. Springer, 1998, pp. 115–132.

[56] Hongfei Fu. "Approximating acceptance probabilities of CTMC-paths on multi-clock deterministic timed automata". In: *HSCC*. 2013, pp. 323–332.

[57] Eli Gafni and Michael Mitzenmacher. "Analysis of Timing-Based Mutual Exclusion with Random Times". In: *SIAM J. Comput.* 31.3 (2001), pp. 816–837.

[58] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees". In: *ACM Trans. Program. Lang. Syst.* 5.1 (1983), pp. 66–77.

[59] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. "CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes". In: *TACAS*. LNCS 6605. Springer, 2011, pp. 372–387.

[60] Flavio D. Garcia, Peter van Rossum, and Ana Sokolova. "Probabilistic Anonymity and Admissible Schedulers". In: *CoRR* abs/0706.1019 (2007).

[61] Sonja Georgievska. "Probability and Hiding in Concurrent Processes". PhD thesis. Eindhoven University, 2011.

[62] Rob Gerth, Ruurd Kuiper, Doron Peled, and Wojciech Penczek. "A Partial Order Approach to Branching Time Logic Model Checking". In: *ISTCS*. 1995, pp. 130–139.

[63] Stephen Gilmore and Jane Hillston. "The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling". In: *Computer Performance Evaluation*. 1994, pp. 353–368.

[64] Rob J. van Glabbeek. "The Linear Time-Branching Time Spectrum I - The Semantics of Concrete, Sequential Processes". In: *Handbook of Process Algebra*. Elsevier, 2001, pp. 3–99.

[65] Rob J. van Glabbeek and Peter Weijland. "Branching Time and Abstraction in Bisimulation Semantics". In: *J. ACM* 43.3 (1996), pp. 555–600.

[66] Rob J. Van Glabbeek, Scott A. Smolka, and Bernhard Steffen. "Reactive, Generative and Stratified Models of Probabilistic Processes". In: *Information and Computation* 121 (1990), pp. 130–141.

[67] Susanne Graf and Hassen Saïdi. "Construction of Abstract State Graphs with PVS". In: *CAV*. LNCS 1254. Springer, 1997, pp. 72–83.

[68] Jan Friso Groote and Frits W. Vaandrager. "An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence". In: *ICALP*. LNCS 443. Springer, 1990, pp. 626–638.

[69] Jan Friso Groote, Aad Mathijssen, Michel A. Reniers, Yaroslav S. Usenko, and Muck van Weerdenburg. "The Formal Specification Language mCRL2". In: *MMOSS*. Dagstuhl Seminar Proceedings 06351. 2006.

[70] Alexander Gruler, Martin Leucker, and Kathrin D. Scheidemann. "Modeling and Model Checking Software Product Lines". In: *FMOODS*. 2008, pp. 113–131.

[71] Dennis Guck, Tingting Han, Joost-Pieter Katoen, and Martin R. Neuhäußer. "Quantitative Timed Analysis of Interactive Markov Chains". In: *NASA Formal Methods*. LNCS 7226. Springer, 2012.

[72] Nicolas Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1992.

[73] Hans Hansson and Bengt Jonsson. "A Logic for Reasoning about Time and Reliability". In: *Formal Asp. Comput.* 6.5 (1994), pp. 512–535.

[74] Holger Hermanns. *Interactive Markov Chains: And the Quest for Quantified Quality*. Springer, 2002. Chap. volume 2428 of LNCS.

[75] Holger Hermanns and Joost-Pieter Katoen. "The How and Why of Interactive Markov Chains". In: *FMCO*. LNCS 6286. Springer, 2009, pp. 311–337.

[76] Charles A. R. Hoare. "Communicating Sequential Processes". In: *Commun. ACM* 21.8 (1978), pp. 666–677.

[77] Charles A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985. ISBN: 0-13-153271-5.

[78] Gerard Holzmann. *Spin Model Checker, The: Primer and Reference Manual*. Addison-Wesley Professional, 2003.

[79] Michael Huth, Radha Jagadeesan, and David A. Schmidt. "Modal Transition Systems: A Foundation for Three-Valued Program Analysis". In: *ESOP*. 2001, pp. 155–169.

[80] Dung T. Huynh and Lu Tian. "On some equivalence relations for probabilistic processes". In: *Fundam. Inform.* 17.3 (1992), pp. 211–234.

[81] "IEEE standard for Property Specification Language (PSL)". In: *IEEE Std 1850TM* (2005).

[82] Wil Janssen. "Layered Design of Parallel Systems". PhD Dissertation. Universiteit Twente, 1994.

[83] Wil Janssen, Mannes Poel, and Job Zwiers. "Action Systems and Action Refinement in the Development of Parallel Systems - An Algebraic Approach". In: *CONCUR*. LNCS 527. Springer, 1991, pp. 298–316.

[84] Wil Janssen and Job Zwiers. "From Sequential Layers to Distributed Processes: Deriving a Distributed Minimum Weight Spanning Tree Algorithm (Extended Anstract)". In: *PODC*. ACM, 1992, pp. 215–227.

[85] Wil Janssen, Mannes Poel, Qiwen Xu, and Job Zwiers. "Layering of Real-Time Distributed Processes". In: *FTRTFT*. LNCS 863. Springer, 1994, pp. 393–417.

[86] Cliff Jones and Gordon D. Plotkin. "A probabilistic powerdomain of evaluations". In: *LICS*. IEEE, 1989, pp. 186–195.

[87] Bengt Jonsson and Kim G. Larsen. "Specification and Refinement of Probabilistic Processes". In: *LICS*. IEEE, 1991, pp. 266–277.

[88] Chi-Chang Jou and Scott A. Smolka. "Equivalences, Congruences, and Complete Axiomatizations for Probabilistic Processes". In: *CONCUR*. LNCS 458. Springer, 1990, pp. 367–383.

[89] Joost-Pieter Katoen, Maneesh Khattri, and Ivan S. Zapreev. "A Markov reward model checker". In: *QEST*. IEEE, 2005, pp. 243–244.

[90] Joost-Pieter Katoen, Daniel Klink, and Martin R. Neuhäußer. "Compositional Abstraction for Stochastic Systems". In: *FORMATS*. LNCS 5813. Springer, 2009, pp. 195–211.

[91] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. "Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking". In: *TACAS*. LNCS 4424. 2007, pp. 87–101.

[92] Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. "Three-Valued Abstraction for Continuous-Time Markov Chains". In: *CAV*. LNCS 4590. Springer, 2007, pp. 311–324.

[93] John George Kemeny and James Laurie Snell. *Finite Markov Chains*. VanNostrand, 1969.

[94] Daniel Klink. "Three-Valued Abstraction for Stochastic Systems". PhD thesis. RWTH Aachen, 2010.

[95] Ron Koymans. "Specifying Real-Time Properties with Metric Temporal Logic". In: *Real-Time Systems* 2.4 (1990), pp. 255–299.

[96] Eyal Kushilevitz and Michael O. Rabin. "Randomized Mutual Exclusion Algorithms Revisited". In: *PODC*. ACM, 1992, pp. 275–283.

[97] Marta Z. Kwiatkowska and Gethin Norman. "Verifying Randomized Byzantine Agreement". In: *FORTE*. LNCS 2529. Springer, 2002, pp. 194–209.

[98] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. "PRISM 2.0: A tool for probabilistic model checking". In: *QEST*. IEEE, 2004, pp. 322–323.

[99] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. "Symmetry Reduction for Probabilistic Model Checking". In: *CAV*. LNCS 4144. Springer, 2006, pp. 234–248.

[100] Marta Z. Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. "Symbolic model checking for probabilistic timed automata". In: *Inf. Comput.* 205.7 (2007), pp. 1027–1077.

[101] Leslie Lamport. "What Good is Temporal Logic?" In: *IFIP Congress*. 1983, pp. 657–668.

[102] Kim G. Larsen. "Modal Specifications". In: *Automatic Verification Methods for Finite State Systems*. LNCS 407. Springer, 1989, pp. 232–246.

[103] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. "On Modal Refinement and Consistency". In: *CONCUR*. LNCS 4703. Springer, 2007, pp. 105–119.

[104] Kim G. Larsen and Arne Skou. "Bisimulation Through Probabilistic Testing". In: *POPL*. 1989, pp. 344–352.

[105] Kim G. Larsen, Bernhard Steffen, and Carsten Weise. "A Constraint Oriented Proof Methodology Based on Modal Transition Systems". In: *TACAS*. LNCS 1019. Springer, 1995, pp. 17–40.

[106] Kim G. Larsen and Bent Thomsen. "A Modal Process Logic". In: *LICS*. IEEE, 1988, pp. 203–210.

[107] Kim G. Larsen and Liu Xinxin. "Equation Solving Using Modal Transition Systems". In: *LICS*. IEEE, 1990, pp. 108–117.

[108] Claire Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. "Property Preserving Abstractions for the Verification of Concurrent Systems". In: *FMSD* 6.1 (1995), pp. 11–44.

[109] Oded Maler, Dejan Nickovic, and Amir Pnueli. "Checking Temporal Properties of Discrete, Timed and Continuous Behaviors". In: *Pillars of Computer Science*. 2008, pp. 475–505.

[110] Robin Milner. *A Calculus of Communicating Systems*. LNCS 92. Springer, 1980.

[111] Robin Milner. "An Algebraic Definition of Simulation Between Programs". In: *IJCAI*. 1971, pp. 481–489.

[112] Robin Milner. "Calculi for Synchrony and Asynchrony". In: *Theor. Comput. Sci.* 25 (1983), pp. 267–310.

[113] Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989.

[114] Yoram Moses and Sergio Rajsbaum. "A Layered Analysis of Consensus". In: *SIAM J. Comput.* 31.4 (2002), pp. 989–1021.

[115] "Mutual Exclusion with Random Times". In: *AVACS S3 Benchmark* (2010).

[116] Shiva Nejati. "Refinement relations on partial specifications". Master thesis. University of Toronto, July 2003.

[117] Martin R. Neuhäußer and Joost-Pieter Katoen. "Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes". In: *CONCUR*. LNCS 4703. Springer, 2007, pp. 412–427.

[118] Rocco De Nicola. "Extensional Equivalences for Transition Systems". In: *Acta Inf.* 24.2 (1987), pp. 211–237.

[119] Rocco De Nicola and Matthew Hennessy. "Testing Equivalences for Processes". In: *Theor. Comput. Sci.* 34 (1984), pp. 83–133.

[120] Rocco De Nicola and Frits W. Vaandrager. "Three Logics for Branching Bisimulation (Extended Abstract)". In: *LICS*. IEEE, 1990, pp. 118–129.

[121] Gethin Norman, D. Parker, M. Kwiatkowska, and S. Shukla. "Evaluating the Reliability of NAND Multiplexing with PRISM". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.10 (2005), pp. 1629–1637.

[122] Ernst-Rüdiger Olderog and Charles A. R. Hoare. "Specification-Oriented Semantics for Communicating Processes". In: *Acta Inf.* 23.1 (1986), pp. 9–66.

[123] Ernst-Rüdiger Olderog and Mani Swaminathan. "Layered Composition for Timed Automata". In: *FORMATS*. LNCS 6246. Springer, 2010, pp. 228–242.

[124] Ernst-Rüdiger Olderog and Mani Swaminathan. "Structural transformations for data-enriched real-time systems". In: *Formal Asp. Comput.* (2014 (to appear)).

[125] Joël Ouaknine and James Worrell. "Some Recent Results in Metric Temporal Logic". In: *FORMATS*. LNCS 5215. Springer, 2008, pp. 1–13.

[126] David Park. "Concurrency and Automata on Infinite Sequences". In: *Proceedings of the 5th GI-Conference on Theoretical Computer Science*. Springer-Verlag, 1981, pp. 167–183.

[127] Augusto Parma and Roberto Segala. "Axiomatization of Trace Semantics for Stochastic Nondeterministic Processes". In: *QEST*. IEEE, 2004, pp. 294–303.

[128] Doron Peled. "Combining Partial Order Reductions with On-the-fly Model-Checking". In: *CAV*. LNCS 818. Springer, 1994, pp. 377–390.

[129] Doron Peled and Thomas Wilke. "Stutter-invariant temporal properties are expressible without the next-time operator". In: *Inf. Process. Lett.* 65.5 (1997), pp. 243–246.

[130] Amir Pnueli. "Linear and Branching Structures in the Semantics and Logics of Reactive Systems". In: *ICALP*. LNCS 194. Springer, 1985, pp. 15–32.

[131] Amir Pnueli. "The Temporal Logic of Programs". In: *FOCS*. 1977, pp. 46–57.

[132] Lucia Pomello. "Some equivalence notions for concurrent systems. An overview". In: *Applications and Theory in Petri Nets*. 1985, pp. 381–400.

[133] Jean-Baptiste Raclet. "Residual for Component Specifications". In: *Electr. Notes Theor. Comput. Sci.* 215 (2008), pp. 93–110.

[134] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, and Roberto Passerone. "Why Are Modalities Good for Interface Theories?" In: *ACSD*. IEEE, 2009, pp. 119–127.

[135] Martin Rem. "Trace Theory and Systolic Computations". In: *PARLE (1)*. Vol. 258. LNCS. Springer, 1987, pp. 14–33.

[136] Michel A. Reniers and Tim A. C. Willemse. "Folk Theorems on the Correspondence between State-Based and Event-Based Systems". In: *SOFSEM*. LNCS 6543. Springer, 2011, pp. 494–505.

[137] Sergey Sazonov. "Property Preservation under Bisimulations on Markov Automata". Master thesis. RWTH Aachen University, August 2013.

[138] David A. Schmidt. "From Trace Sets to Modal-Transition Systems by Stepwise Abstract Interpretation". In: 2001, pp. 53–71.

[139] Roberto Segala. "Modelling and Verification of Randomized Distributed Real Time Systems". PhD thesis. MIT, 1995.

[140] Roberto Segala. "Testing Probabilistic Automata". In: *CONCUR*. LNCS 1119. Springer, 1996, pp. 299–314.

[141] Roberto Segala and Nancy A. Lynch. "Probabilistic Simulations for Probabilistic Processes". In: *Nord. J. Comput.* 2.2 (1995), pp. 250–273.

[142] Arpit Sharma. "A Two Step Perspective for Kripke Structure Reduction". In: *CoRR* abs/1210.0408 (2012).

[143] Arpit Sharma. "Weighted Probabilistic Equivalence Preserves Omega-Regular Properties". In: *MMB/DFT*. 2012, pp. 121–135.

[144] Arpit Sharma and Joost-Pieter Katoen. "Layered Reduction for Abstract Probabilistic Automata". In: *ACSD*. IEEE, 2014 (to appear).

[145] Arpit Sharma and Joost-Pieter Katoen. "Layered Reduction for Modal Specification Theories". In: *FACS*. 2013, pp. 329–347.

[146] Arpit Sharma and Joost-Pieter Katoen. "Weighted Lumpability on Markov Chains". In: *Ershov Memorial Conference (PSI)*. Vol. 7162. LNCS. Springer, 2012, pp. 322–339.

[147] Falak Sher and Joost-Pieter Katoen. "Compositional Abstraction Techniques for Probabilistic Automata". In: *IFIP TCS*. LNCS 7604. Springer, 2012, pp. 325–341.

[148] Jan L. A. van de Snepscheut. *Trace Theory and VLSI Design*. Vol. 200. LNCS. Springer, 1985.

[149] Mani Swaminathan, Joost-Pieter Katoen, and Ernst-Rüdiger Olderog. "Layered reasoning for randomized distributed algorithms". In: *Formal Asp. Comput.* 24.4-6 (2012), pp. 477–496.

[150] Wolfgang Thomas and Thomas Wilke. *Automata Logics, and Infinite Games: A Guide to Current Research*. LNCS 2500. Springer-Verlag, 2002.

[151] Chris M. N. Tofts. "Compositional Performance Analysis". In: *TACAS*. LNCS 1217. Springer, 1997, pp. 290–305.

[152] Chris M. N. Tofts. "Processes with Probablities, Priority and Time". In: *Formal Asp. Comput.* 6.5 (1994), pp. 536–564.

[153] Sebastián Uchitel and Marsha Chechik. "Merging partial behavioural models". In: *SIGSOFT FSE*. ACM, 2004, pp. 43–52.

[154] Antti Valmari and Giuliana Franceschinis. "Simple O(m log(n)) Time Markov Chain Lumping". In: *TACAS*. LNCS 6015. 2010, pp. 38–52.

[155] Simone Veglioni and Rocco De Nicola. "Possible Worlds for Process Algebras". In: *CONCUR*. LNCS 1466. Springer, 1998, pp. 179–193.

[156] Björn Wachter, Lijun Zhang, and Holger Hermanns. "Probabilistic Model Checking Modulo Theories". In: *QEST*. IEEE, 2007, pp. 129–140.

[157] Verena Wolf, Christel Baier, and Mila E. Majster-Cederbaum. "Trace Machines for Observing Continuous-Time Markov Chains". In: *ENTCS* 153.2 (2006), pp. 259–277.

[158] Verena Wolf, Christel Baier, and Mila E. Majster-Cederbaum. "Trace Semantics for Stochastic Systems with Nondeterminism". In: *Electr. Notes Theor. Comput. Sci.* 164.3 (2006), pp. 187–204.

[159] Christoph Worreschk. "Weighted Lumpability on Markov chains". Bachelor thesis. RWTH Aachen University, August 2012.

[160] Lijun Zhang and Martin R. Neuhäußer. "Model Checking Interactive Markov Chains". In: *TACAS*. LNCS 6015. 2010, pp. 53–68.

Curriculum Vitae

Arpit Sharma was born on the 2nd of September 1984 in Pilani, India. He studied computer science at the University of Rajasthan, Jaipur, India and obtained the degree of Bachelor in Computer Science and Engineering in May, 2006. In September 2009 he obtained a M.Tech. degree in Software Engineering from Manipal Institute of Technology, India and a M.Sc. degree in Computer Science from Eindhoven University of Technology, The Netherlands. Since June 2010 he is a Ph.D. student at the Software Modeling and Verification Group, Department of Computer Science, RWTH Aachen University, Germany.